Chapter 6

Fighting the Sampling Bias: A Framework for Training and Evaluating Credit Scoring Models

Publication

Kozodoi, N., Lessmann, S., Alamgir, M., Gatsoulis, Y., Moreira-Matias, L., & Papakonstantinou, K. (2021). Fighting the Sampling Bias: A Framework for Training and Evaluating Credit Scoring Models. Submitted to Management Science (2nd review round).

Abstract

Scoring models support decision-making in financial institutions. Their estimation relies on the data of previously accepted applicants with known repayment behavior. This creates sampling bias: the training data offers a partial picture of the distribution of candidate borrowers to which the model is applied when screening new applications. The paper makes two contributions to address the adverse effect of sampling bias on model evaluation and training. First, we propose a Bayesian evaluation framework that extends standard evaluation metrics to the biased setting and provides a reliable estimate of future scorecard performance. To improve training, we develop Bias-aware self-learning – a reject inference framework that augments the biased training data by inferring labels for selected rejected applications. Extensive experiments on synthetic and real-world data confirm the superiority of our propositions over previous bias correction methods in terms of predictive performance and profitability and identify boundary conditions affecting their performance.

6.1 Introduction

The rise of big data and AI impacts management practices and decision processes in the financial industry. Financial institutions use scoring models to support resource allocation, inform risk management, and automate operational decision processes. A scoring model predicts the future state of a variable based on observational data. Credit scorecards are a prominent example. Estimating a borrower's probability to default, they support loan approval decisions and loss provisioning [11]. Generally speaking, the value of a scoring model depends on its ability to generate accurate predictions when processing new data not seen during model development [40]. We examine the practices underneath scorecard construction and argue that these create sampling bias, which diminishes the quality of scorecard-based decisions.

Application scorecards, which estimate an applicant's repayment ability, illustrate the problem. To obtain the data required for scorecard estimation, a financial institution labels previous loan applications with a known outcome according to whether a debt was repaid or a default event occurred. We refer to the corresponding applications as *good* or *bad* risks. Class labels are observed for previously granted applications. Inevitably, the sample of accepted clients differs from the overall population of applicants, which includes applicants the scorecard would reject. Lacking the labels of rejected clients creates a missing data problem. Approving applications using a scorecard implies that application labels of rejected clients are either missing at random (MAR) or not at random (MNAR), which leads to sampling bias [62]. The bias impedes model training and evaluation. Training a scorecard on data from a biased sample may deteriorate the accuracy of its predictions when the model is used to screen new applications. Evaluating a model on a biased sample provides a misleading estimate of its actual performance.

The prevalence of scorecard-based decisions warrants concern about the sampling bias. In 2021, the total outstanding amount of consumer credit in the US exceeded \$4,325 billion[1]. Scorecards played a major role in the approval of this amount of credit. Given a trend to attain financing via financial technology companies (FinTechs), we expect the importance of scoring models to increase even further. Many FinTechs rely on a data-driven business model and the automation of loan approval. Thus, risk scores produced by scoring models increasingly determine access to finance, which plays a crucial role in economic inequality [101] and extends the impact of sampling bias beyond the accuracy of individual approval decisions. Applications of conceptually similar models to inform, for example, corporate lending [43] and the management of mortgage portfolios [82], corroborate this view. The availability of labeled data is crucial to supervised machine learning (ML), making sampling bias a serious concern in an increasingly data- and model-driven economy.

The goal of the paper is to shed light on the severity of sampling bias and develop strategies to mitigate its adverse effect on the two key steps of an ML pipeline, training and evaluation. Our first contribution is a new evaluation framework for scorecard assessment. Traditional performance measures such as, e.g., the area under a receiver operating characteristics curve (AUC), require labeled data. The labels are not available for rejected clients. Assessing a scorecard on accepts provides a misleading performance estimate. Reliable model validation is important for judging the model's business value, informing long-term planning and risk assessment decisions as well as performing the model selection. We propose a Bayesian evaluation framework that allows calculating an arbitrary performance measure on a representative sample from the borrowers' population that includes rejects. Drawing on prior knowledge, our framework avoids dependence on the actual labels of rejects and facilitates accurate evaluation under sampling bias.

Second, we introduce bias-aware self-learning (BASL) – a reject inference framework

[1]Source: The Federal Reserve (2021) Statistical Release on Consumer Credit, `https://www.federalreserve.gov/releases/g19/current`.

that mitigates the impact of sampling bias on scorecard performance. BASL augments the training data by labeling selected rejected cases and comprises procedures to address the high uncertainty associated with label estimation. For example, we establish the importance of involving learning algorithms with different characteristics – strong and weak learners – and propose a filtering stage to restrict the labeling to a suitable subset of rejected applications. The BASL framework extends our previous work on reject inference [55].

We test our propositions on synthetic and real-world data. First, we set up a controllable synthetic environment in which the labels of rejects are known and develop a data generation algorithm that mimics the loan approval cycle supported by a scoring model. The simulation study illustrates sampling bias and its adverse impact on the scorecard training and evaluation. It also allows us to investigate boundary conditions that influence the magnitude of the loss due to bias and the performance gains from our propositions. Second, we compare the proposed methods to established bias correction benchmarks on a real-world high-dimensional microloan data set. The data set includes a sample of applications that were randomly accepted without scoring. This sample represents the operating conditions of a scorecard and uncovers the true merit of bias correction [26]. The unbiased sample allows us to evaluate the performance of the proposed methods properly and measure performance gains in monetary terms.

It is worth noting that each of the two contributions of the paper can be used on a standalone basis. The first contribution ensures that scorecards are evaluated in a suitable way when sampling bias is present. The second contribution represents a reject inference framework that supports any supervised ML algorithm and can improve its performance under sampling bias. The two contributions combined constitute a holistic approach to sampling bias mitigation in credit scoring.

6.2 Theoretical Background

This section formalizes the sampling bias problem in credit scoring in relation with the missingness mechanisms. Let $X \in \mathbb{R}^k$ denote a loan applicant. The matrix of the applicants' attributes is denoted as $\mathbf{X} = (X_1, ..., X_n)^\top$, and $\mathbf{y} = (y_1, ..., y_n)^\top$ is a random vector of binary labels, indicating if the applicant repays the loan ($y = 0$) or defaults ($y = 1$). Suppose $\mathbf{X}$ and $\mathbf{y}$ have marginal distributions denoted as $\mathbf{P}_X$ and $\mathbf{P}_Y$ and a joint distribution $\mathbf{P}_{XY} = \mathbf{P}(y|X)$. Given a set of independent and identically distributed applications $D = \{(\mathbf{X}, \mathbf{y})\}$ with $(\mathbf{X}, \mathbf{y}) \sim \mathbf{P}_{XY}$, a financial institution uses a scorecard $f(X)$ that approximates $\mathbf{P}(y = 1|X)$ to split D into two subsets: accepts D^a and rejects D^r, $D = D^a \sqcup D^r$. The repayment behavior is eventually observed for applicants in D^a, while the labels of rejects remain unknown. In other words, D exhibits missingness with respect to $\mathbf{y}$.

The missingness mechanism has implications for credit scorecards. Let $\mathbf{a} \in \{0, 1\}$ denote a binary variable indicating if the applicant's repayment outcome is observed ($a = 1$) or missing ($a = 0$), which corresponds to whether the applicant was accepted. Labels are missing completely at random (MCAR) if $\mathbf{P}(a|X,y) = \mathbf{P}(a)$, implying that missingness is not related to the data and no bias correction is needed. A finite-sample bias, which may occur due to limited sample size, can be reduced by collecting more data [6]. In credit scoring, MCAR occurs only if a bank accepts applications at random, which is unrealistic and does not warrant further consideration.

Filtering accepts using a scorecard causes D^a to have different empirical distributions compared to $\mathbf{P}_{XY}$, $\mathbf{P}_X$ and $\mathbf{P}_Y$ and creates sampling bias. We face MAR if $\mathbf{P}(a|X,y) = \mathbf{P}(a|X)$, which implies that the label missingness does not depend on the repayment status and is driven by the applicants' attributes $\mathbf{X}$. This occurs if a financial institution does not use any external information apart from $\mathbf{X}$ to make acceptance decisions (e.g., always relies on predictions of the same scorecard). Under MAR, posterior probability models such as logistic regression (LR) trained on a biased sample produce unbiased estimates and do not require bias correction [8]. However, the performance of certain classifiers may deteriorate. This concerns tree-based models that split the training data based on the observed feature values and, therefore, fail to extrapolate on new examples that lie outside of the previously observed feature ranges [71]. In credit scoring, tree-based classifiers such as random forest (RF) or extreme gradient boosting (XGB) were shown to outperform other benchmarks [e.g., 41, 57]. Using such models for scorecard development emphasizes the need for sampling bias correction in the MAR setting.

The MNAR setting is more challenging and implies that missingness depends on $\mathbf{y}$ due to unobserved factors that can not be explained through the attributes $\mathbf{X}$. Formally, the data exhibits MNAR if $\mathbf{P}(a|X,y) \neq \mathbf{P}(a|X)$. In practice, it is difficult to distinguish MNAR and MAR since the unobserved factors might not be accessible. In credit scoring, one of the main drivers of MNAR is manual overwriting of the scorecard predictions based on attributes not included in $\mathbf{X}$, which ties missingness to the factors unknown to the model $f(X)$. For instance, applicants with a County Court Judgment may be manually rejected by a decision-maker even if the scorecard prediction is positive [8]. MNAR can also occur when some of the features in $\mathbf{X}$ included in a previous scorecard can no longer be used by a financial institution (e.g., due to new data privacy regulations or changes in data providers). MNAR leads to biased model parameters [42, 62], which harms the performance of a model trained on a biased sample. The bias correction under MNAR is needed irrespective of the base classifier.

Apart from impacting model training, sampling bias adversely affects model evaluation under both MAR and MNAR. A validation subset H^a drawn from the labeled set D^a is not representative of D if the labels do not exhibit MCAR. As a result, evaluating $f(X)$ on a subset of previously accepted applicants will provide misleading performance estimates with regards to the actual performance of $f(X)$ on new loan applications drawn from $\mathbf{P}_{XY}$. In credit scoring, D^a contains applications predicted as least risky, which usually leads to overoptimistic performance estimates when using accepts-based evaluation [9].

We formalize the research goal of this paper as follows: given a set of labeled accepts D^a and unlabeled rejects D^r, whereby labels are not MCAR, we strive to: (i) infer a function $f(X)$ that approximates $\mathbf{P}(y = 1|X)$ and generalizes well over applications from $\mathbf{P}_{XY}$ and (ii) estimate the predictive performance of $f(X)$ over applications from $\mathbf{P}_{XY}$. The task aims at improving the scorecard performance and the accuracy of estimates of scorecard performance, respectively. Exploiting the information in D^r can help to reduce the impact of sampling bias in both tasks.

6.3 Related Work

Sampling bias has received much attention in the literature. Prior work considers missing data problems when some examples are not observed due to non-random sampling [e.g., 25]. A related concept is domain adaptation, or data set shift, which studies differences in the training and test distributions due to, for example, a shift in the marginal feature distributions [88]. Model training and evaluation on biased incomplete data is also considered in the literature on off-policy learning and evaluation [31, 4]. This section reviews prominent bias correction methods and empirical studies on sampling bias in credit scoring. A survey of bias correction techniques is available in Table 6.9.1 in Appendix 6.9.1.

6.3.1 Training under Sampling Bias

Representation change is a family of bias correction methods applied in the data prepossessing stage before training a corrected model. Such methods assume MAR and use a mapping function Φ to project features into a new representational space $\mathbf{Z}$, $\Phi : \mathbf{X} \rightarrow \mathbf{Z}$, such that the training data distribution over $\mathbf{Z}$ is less biased and $\Phi(X)$ retains as much information about X as possible. A suitable representation is found by maximizing a distribution similarity measure such as the distance between the distribution moments in a kernel space [15] or during feature selection and/or transformation [e.g., 23, 80].

A recent feature transformation approach trains a deep autoencoder with a mismatch penalty and extracts the corrected data representation from the bottleneck layer [3]. Using such transformation harms model comprehensibility, whereas regulatory compliance requires financial institutions to ensure comprehensible scoring models [5].

Model-based bias correction methods modify a learning algorithm to account for the bias. In his pioneering work, Heckman [42] proposed a two-stage least-squares model for the MNAR setup. The Heckman model simultaneously estimates two equations: the outcome and the sample selection process, which allows to eliminate bias in the estimated model parameters and yield consistent estimates. Building on the linear Heckman model, Meng and Schmidt [77] developed a bivariate probit model with non-random sample selection for setups where the outcome variable is binary. Their model represents a theoretically sound approach for the credit scoring setup under assumptions of MNAR and normally distributed residuals in the estimated equations.

Another research stream considers mixture models for bias correction [e.g., 32]. Mixture models operate under the MAR assumption and treat the data as drawn from a mixture of two distributions: training and population. Learning from the labeled training sample and unlabeled sample from the population, such models infer labels of new examples using the conditional expectation-maximization algorithm for maximum likelihood estimation.

The main disadvantage of model-based methods is that they are embedded in a learning algorithm, which requires a specific classifier. Previous work has mostly focused on linear and parametric models with particular assumptions. Yet, there is evidence that other non-parametric algorithms such as XGB demonstrate better performance in credit scoring [e.g., 41].

Reweighting is another method that rebalances the training loss towards representative examples. Weights of the training examples, also known as importance weights or propensity scores, can be computed as a ratio of the two distribution densities: $w(X) = p_D(X)/p_{D^a}(X)$. High values of $w(X)$ indicate that X is more likely drawn from $\mathbf{P}_{XY}$ and is, therefore, more important for training. Prior work suggests numerous techniques for importance weight estimation. For example, a model-based method estimates weights by fitting a classifier $c(X)$ on D using a binary sample indicator s as a label, where $s(X) = 1$ if $X \in D^a$ and 0 otherwise. Kernel Mean Matching [45] estimates density ratios by matching distributions in kernel space. Another idea is to use cluster-based empirical frequencies by splitting the data into clusters and computing weights as a ratio of test and training examples within clusters [25]. The importance weights can then be used during scorecard training using, for example, weighted least squares.

Since reweighting only relies on attributes in $\mathbf{X}$, it assumes MAR and can not correct for MNAR. However, reweighting can still be helpful under MNAR as it may reduce error in estimating a model from the training sample [8]. Another limitation of reweighting is that it faces difficulties in high-dimensional feature spaces where weight estimates exhibit high variance [100]. Last, a reweighted training set still consists of previously accepted clients and misses certain distribution regions populated by rejects only.

The credit scoring literature has also explored the idea of data augmentation – expanding the training sample by labeling and appending examples from D^r. The augmented sample covers a wider distribution region, which reduces sampling bias. Prior work suggests different approaches that use a model trained over D^a to label rejects. A classic example is hard cutoff augmentation (HCA), which labels rejects by comparing their scores predicted with the accepts-based model to a predefined threshold. Under sampling bias, reliance on the accepts-based model may increase the risk of error propagation when labeling rejects. Extrapolating predictions of the accepts-based scorecard on rejects is, therefore, a valid technique for posterior probability classifiers under MAR but suffers from the omitted variable bias under MNAR [9].

Parceling aims to improve upon HCA by considering rejects as riskier than accepts. Parceling splits rejects into segments based on the predicted score range and labels rejects within each range proportional to the assumed probability of default in that range. The probabilities can then be altered by a decision-maker compared to the ones observed within the same score range on D^a. This implies that parceling can work in MNAR settings if the decision-maker is able to correctly specify the change in the default probabilities across the considered groups of applicants.

This paper introduces BASL – a reject inference framework that builds on self-learning based data augmentation and incorporates important extensions to account for the presence of sampling bias. The framework is model-agnostic and includes distinct regimes for labeling rejects and training a resulting scorecard. This allows us to reduce the risk of error propagation during labeling rejects and employ a classifier with high discriminative power for screening new applications.

6.3.2 Evaluation under Sampling Bias

The difference in data distributions also affects model evaluation. An estimate of model performance derived from a biased sample may not generalize to unseen data, which causes standard model evaluation strategies such as cross-validation to fail [91].

To address the evaluation problem, prior work proposes generalization error measures, whose asymptotic unbiasedness is maintained under sampling bias. This includes the modified Akaike information criterion [MAIC, 86] and the generalization error measure suggested by Sugiyama et al. [92]. Both measures rely on density ratio estimation to compensate for the distribution differences between the training and the test data and can thus be less accurate in high-dimensional feature spaces. Measures like the MAIC are also limited to parametric learners.

Evaluation under sampling bias is also studied in off-policy evaluation research, which focuses on the evaluating of a policy (e.g., a classifier) in a contextual bandit setting with incomplete historical data [87]. In this setup, a policy reward depends on the action of a decision-maker and is only partially observed. A prominent policy evaluation method is importance-weighted validation, which reweights the reward towards more representative examples in the evaluation set using importance weights [91]. In a binary classification setting, policy reward corresponds to an assessment of the scoring model performance using some evaluation metric.

Reweighting produces biased estimates if the past policy is modeled incorrectly [31]. In our setting, this implies that the attributes in $\mathbf{X}$ do not explain previous acceptance decisions accurately, and the data exhibits MNAR. For such cases, Dudik et al. [31] recommend doubly robust (DR) estimators, which combine estimating importance weights with predicting policy reward (i.e., classifier loss). DR produces unbiased estimates if at least one of the modeled equations is correct. However, using DR in credit scoring is difficult. The contextual bandit setting considers a set of actions to decide on a case and assumes that we observe a reward for one of those actions. DR can then impute the reward for other actions. In credit scoring, however, we do not observe a reward for rejected clients, which complicates the imputation of reward substantially. Also, measuring reward as classifier loss limits DR to performance measures calculated on the level of an individual loan. This prohibits using DR with rank-based metrics such as the area under the ROC curve (AUC), which are established in credit scoring [e.g. 82].

This paper introduces a Bayesian evaluation framework that remedies the adverse impact of sampling bias on model evaluation and provides a more reliable estimate of model performance. The framework is metric-agnostic and allows evaluating any scoring model on a data sample with labeled accepts and unlabeled rejects. The framework leverages prior knowledge of the label distribution among rejects and uses Monte-Carlo sampling to optimize calculations.

6.3.3 Applications in Credit Scoring

Sampling bias has gained considerable attention in credit scoring. Previous research has mostly focused on the impact of sampling bias on scorecard training and tested some bias correction techniques including the Heckman model [e.g., 9], data augmentation techniques such as HCA [e.g., 26], and mixture models [e.g., 32]. A commonly used reweighting approach is banded weights, a cluster-based method that uses the bands of predicted probabilities of default to form clusters [7]. Recent studies also explore semi-supervised learning methods such as self-learning and semi-supervised SVMs [e.g., 59]. Several studies conclude that gains from reject inference are little or non-existent [7, 21]. At the same time, only a few studies express performance gains in terms of profitability [e.g., 21] or use a proper representative sample to measure gains from bias correction [e.g., 8]. Table 6.9.2 in the Appendix provides a detailed overview of empirical studies on reject inference.

The problem of evaluation under sampling bias has received less attention in the credit scoring literature. This can be attributed to limited data availability. Analyzing the impact of bias on evaluation requires a representative holdout sample that includes labeled applicants rejected by a scorecard. Seven out of 25 studies presented in Table 6.9.2 have (partial) access to the labels of some of the real rejects. However, two of these studies focus on corporate credit scoring, and the remaining five rely on just two distinct consumer credit scoring data sets. Using such proprietary data, Banasik et al. [9] illustrate the discrepancies between the accuracy estimates obtained on a sample from accepts and a representative holdout sample and use the latter to judge the value of reject inference. To the best of our knowledge, previous work has not considered techniques that aim to correct the impact of sampling bias on model evaluation in the absence of such a sample.

Another limitation of empirical studies on reject inference is that the employed data sets are usually low-dimensional (see Table 6.9.2). While traditional banks still rely on parsimonious scorecards, this is not typical for FinTechs, which operate with large amounts of high-dimensional data from different sources [89]. Recent studies also indicate that financial institutions increasingly rely on alternative data such as applicants' digital footprints, e-mail activity and others [e.g. 12]. This trend emphasizes the importance of coping with high-dimensional data in reject inference.

This paper aims to address limitations of the prior work on sampling bias in credit scoring by employing a high-dimensional FinTech data set, evaluating performance on a representative sample from the borrowers' population, and examining the business impact of reject inference.

6.4 Bayesian Evaluation Framework

Estimating the performance of a scorecard before applying the model to screen new loan applications is a crucial task for decision-makers. An accurate estimate of the future model performance is necessary for judging the business value of the model, informing long-term planning and risk assessment decisions, and selecting a model variant expected to achieve better performance. Evaluating a model on a biased validation set of previously accepted applicants produces biased performance estimates. As a result, the actual scorecard performance in production does not match the expectations raised in the model evaluation stage. This section introduces the Bayesian evaluation framework that aims at mitigating the adverse effect of sampling bias on performance evaluation.

6.4.1 Evaluation Framework

Recall that we are given a population of loan applicants $D = D^a \sqcup D^r$, where D^a is accepts and D^r is rejects. Let $f(X)$ denote the scoring model to be evaluated. To infer its true ability to assess the creditworthiness of new applicants, $f(X)$ has to be evaluated on a representative holdout set denoted as $H, H \subset D$ [9]. Calculating standard evaluation measures would require knowledge of labels for all cases in H. However, in practice, only the labels in $H^a = H \cap D^a$ are known. The labels of rejects in $H^r = H \cap D^r$ are not available. A common approach, called accepts-based evaluation hereinafter, is to assess $f(X)$ based on H^a. Empirical results in Section 6.7 illustrate the inappropriateness of this approach and emphasize the need for improvement.

The proposed Bayesian framework extends standard performance metrics by incorporating rejects and available information on their label distribution. Assume $f(X)$ is evaluated

```
  input  : model $f(X)$, evaluation set $H$ consisting of labeled accepts
           $H^a = \{(\mathbf{X}^a, \mathbf{y}^a)\}$ and unlabeled rejects $H^r = \{\mathbf{X}^r\}$, prior $\mathbf{P}(\mathbf{y}^r|\mathbf{X}^r)$,
           evaluation metric $\mathrm{M}(f, H, \tau)$, meta-parameters $j_{max}, \varepsilon$
  output: Bayesian evaluation metric $\mathrm{BM}(f, H, \tau)$
1 $j = 0$; $\Delta = \varepsilon$; $E = \{\}$ ;                        // initialization
2 while $(j \leq j_{max})$ and $(\Delta \geq \varepsilon)$ do
3     $j = j + 1$
4     $\mathbf{y}^r = \text{binomial}(1, \mathbf{P}(\mathbf{y}^r|\mathbf{X}^r))$ ;       // generate labels of rejects
5     $H_j = \{(\mathbf{X}^a, \mathbf{y}^a)\} \cup \{(\mathbf{X}^r, \mathbf{y}^r)\}$ ;    // construct evaluation sample
6     $E_j = \frac{1}{j} \sum_{i=1}^{j} \mathrm{M}(f, H_i, \tau)$ ;                    // evaluate f(X)
7     $\Delta = E_j - E_{j-1}$ ;                         // check metric convergence
8 end
  return: $\mathrm{BM}(f, H, \tau) = E_j$
```

Algorithm 3: Bayesian Evaluation Framework

on H using an arbitrary evaluation metric $\mathrm{M}(f, H, \tau)$, where τ is a vector of metric meta-parameters (e.g., classification cut-off). Algorithm 3 computes the Bayesian extension of the metric M denoted as $\mathrm{BM}(f, H, \tau)$. Since the labels of examples in H^r are unknown, we choose a prior of the label distribution among rejects $\mathbf{P}(\mathbf{y}^r, \mathbf{X}^r)$ and assign random pseudo-labels accordingly. This allows us to evaluate $f(X)$ on a representative sample consisting of labeled accepts and pseudo-labeled rejects.

Within the Bayesian evaluation framework, we employ Monte-Carlo sampling to optimize computation. Each unknown label is drawn from a binomial distribution with the probability set to the prior for that rejected example. The Bayesian extension of the metric is then computed by averaging the metric values across multiple label realizations. The sampling iterations are terminated once the incremental change of the average value does not exceed a convergence threshold ε.

As commonly observed in Bayesian estimation, the accuracy of the estimated metric depends on the choice of the prior. Instead of using the less informative and difficult to estimate class prior $\mathbf{P}(\mathbf{y}^r)$, we recommend leveraging the attributes of the cases in D^r denoted by $\mathbf{X}^r$. This enables estimating the prior $\mathbf{P}(\mathbf{y}^r|\mathbf{X}^r)$ by rescoring rejects with a model that has been used to support loan approval decisions in the past or use the original scores used in those decisions if they are available. The prior governs the sampling of class labels of rejects. In Section 6.7.1, we show that the scores of the past model should display high calibration. It is also essential that the past model has been trained on the historical data that is not part of the evaluation set H.

Note that a model trained on accepts to compute the prior also suffers sampling bias. The proposed evaluation framework stands on the trade-off of two components: the benefit from evaluating a model on a larger sample more representative of the population and the noise in the simulated labels of rejects. As we establish through empirical experimentation and illustrate in Section 6.7.1, gains from extending the evaluation sample outweigh losses from the noise in the prior, which facilitates a good performance of the Bayesian framework.

6.4.2 Applications to Performance Metrics

One of the key advantages of the proposed evaluation framework is its support of arbitrary performance metrics. When evaluating credit scorecards, it is beneficial to take into account not only the operating conditions, such as the class priors and the misclassification costs, but also the financial institution's strategy, such as risk minimization, growth, or profit maximization. This ensures that the institution's objectives can be met by mapping the evaluation output to the company's key performance indicators and that all relevant information is used. The model-agnostic nature of the proposed framework allows it to benefit any institution irrespective of the chosen metric.

The AUC and the Brier Score (BS) are widely used evaluation measures in credit scoring that do not have any meta-parameters. The AUC is an established indicator of the discriminatory ability of a scorecard. Calculating the mean squared error between model-estimated predictions and a zero-one coded default indicator, the BS measures the degree to which the model predictions are well-calibrated. Under sampling bias, estimates of the AUC and BS on a sample from D^a will be misleading as D^a only represents a limited region of the target data distribution. A Bayesian extension of each of these two metrics can be computed on a holdout set H using Algorithm 3.

In credit scoring, accepting a *bad* applicant incurs higher costs than rejecting a *good* applicant. The Partial AUC (PAUC) summarizes the ROC curve on a limited range of thresholds and facilitates accounting for asymmetric error costs [99]. A financial institution can measure the PAUC to evaluate the ranking ability of a model in the area of the ROC curve with a low false negative rate (i.e., FNR $\in [0, \xi]$). The upper bound ξ is a meta-parameter of the metric and limits the range of thresholds to ensure that the acceptance rate is sufficiently low such that the target FNR is not exceeded. Similar to the AUC, we can estimate the Bayesian PAUC on H using Algorithm 3. In this paper, we compute the PAUC in the area of the ROC curve with FNR $\in [0, .2]$.

Assuming that a financial institution's objective is to maximize the acceptance rate while keeping credit losses below a certain threshold or to minimize credit losses while approving a specific percentage α of loan applications, a suitable evaluation metric is the *bad* rate among accepts (ABR), where accepts are the top $\alpha\%$ applications with the lowest estimated probabilities of default. Computing the ABR on H^a leads to an over-optimistic performance estimate since H^a already represents the top $\alpha\%$ applications from the population. The Bayesian ABR computed on H using the proposed evaluation framework provides a more reliable estimate of the *bad* rate among accepts. In this paper, we integrate the ABR over acceptance between 20% and 40%, which is a historical acceptance range on the real-world lending data set used in the paper.

6.5 Bias-Aware Self-Learning Framework

Training a scorecard on a biased sample results in a performance loss when the model is applied to screen new applications from D. This section introduces a reject inference framework aimed at mitigating the impact of sampling bias on training. We start by revisiting traditional self-learning that serves as a base for BASL and extend it to a setup where the data exhibits sampling bias.

6.5.1 Traditional Self-Learning

Self-learning is an incremental semi-supervised learning framework [58]. Given a labeled set D^a and an unlabeled set D^r, self-learning trains a supervised model over D^a. Next, the trained model is used to score examples in D^r. Examples where model predictions exceed the specified confidence thresholds are assigned the corresponding labels and appended to D^a. The classifier is then retrained on the augmented labeled sample to score the remaining unlabeled data. The procedure is repeated until a stopping criterion is met.

Sampling bias impedes the effectiveness of self-learning. First, labeling rejects based on the confident predictions of a model trained on accepts may be misleading. Since rejects come from a different distribution region, the accepts-based model can produce overconfident predictions that become less reliable as the difference between the two samples increases. The prediction errors are propagated through consecutive labeling iterations, impairing the resulting performance. The accuracy of the assigned labels is further threatened when using a strong learner, which may be prone to overfitting the biased training sample. Using the same confidence thresholds for labeling *good* and *bad* rejects also results in preserving the class ratio in the augmented labeled sample, whereas the *bad* ratio on a representative sample of loan applicants is expected to be higher than on accepts. Finally, employing commonly used stopping criteria based on the absence of examples with confident predictions may lead to exceeding the suitable number of labeling iterations, which risks overfitting the sample of accepts and can strengthen the error propagation due to the bias.

6.5.2 Bias-Aware Self-Learning

The proposed BASL framework addresses the limitations of traditional self-learning and extends it to a setup where the data is missing not completely at random. The extensions include: (i) introducing a filtering stage before labeling; (ii) implementing modifications to the labeling stage and training regime; (iii) introducing stopping criteria to handle sampling bias. The BASL framework is visualized in Figure 6.5.1. The pseudo-code is provided in Algorithm 6 in Appendix 6.9.2.

Note that BASL does not aim to solve the fundamental extrapolation problem by completely eliminating bias in the training data. This is not feasible as the repayment behavior

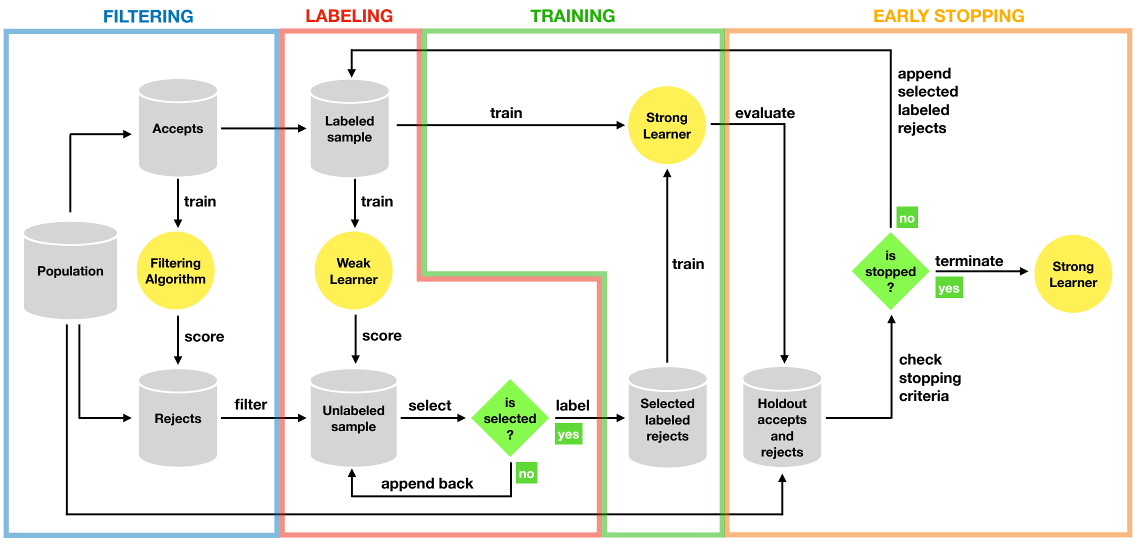

Figure 6.5.1. Bias-Aware Self-Learning Framework

Population is partitioned into labeled accepts, unlabeled rejects and a holdout set containing accepts and rejects. Filtering algorithm filters rejects before labeling; we use isolation forest trained on accepts. Weak learner assigns labels to rejects; we use LR. Strong learner screens new loan applications; we use XGB. Selection criteria: (i) random sample of rejects with a rate ρ; (ii) scores of weak learner exceed confidence thresholds based on percentile thresholds γ and imbalance multiplier θ. Stopping criteria: (i) reaching maximum no. iterations j_{max}, (ii) performance of strong learner measured with the Bayesian framework is not improving; (iii) set of selected labeled rejects is empty.

of rejects from a too distant distribution region is hard to estimate. Therefore, we ensure that the training data is only augmented with a few rejects, for which the labeling model is confident and the data distribution is not too different from accepts. By doing so, we can not expect the augmented sample to be free of sampling bias. Instead, we aim at reducing the bias while keeping the error propagation sufficiently low. The trade-off between bias reduction and scorecard accuracy is a crucial element of the BASL framework. Appendix 14 further investigates this trade-off by constructing bias-accuracy Pareto frontiers with BASL variants that label different subsets of rejects.

Filtering Stage

In the first stage, we filter rejects in D^r. The goal of the filtering is two-fold. First, we aim at removing rejects coming from the most different part of distribution compared to D^a. Removing such examples reduces the risk of error propagation since predictions of a model trained over accepts become less reliable as the distribution of examples to be labeled shifts further from the one observed during training. Second, we aim at removing rejects that are most similar to accepts. Labeling such examples would potentially provide little new information for a scorecard and might even harm performance due to introducing noise.

We estimate the similarity between rejects and previously accepted applications by scoring rejects in D^r with a novelty detection algorithm trained over D^a. To remove the rejects most and least similar to accepts, we drop examples within the top β_u and bottom β_l percentiles of the predicted similarity scores. The threshold values $\beta = (\beta_u, \beta_l)$ act as meta-parameters of the filtering algorithm, which we implement using isolation forest, a scalable tree-based novelty detection algorithm suitable for high-dimensional feature spaces [65].

Labeling Stage

After filtering, we iteratively label selected rejects. We employ distinct regimes for labeling rejects and training the resulting scorecard and suggest scoring rejects using a learner with different inductive bias compared to the one employed for scorecard construction. The labeling algorithm should provide well-calibrated predictions to select the appropriate confidence thresholds. Another desideratum of the labeling algorithm is that it should be less prone to overfitting the biased training sample. Using different algorithms for reject inference and scoring new applications also reduces the risk of amplifying the bias of the base classifier.

We use L1-regularized LR as a weak learner for labeling rejects. The L1 penalty is introduced when working with high-dimensional data with noisy features. LR is a parametric learner that outputs probabilistic predictions. As we show in Appendix 6.9.2, predictions of LR are better calibrated and take extreme values less frequently compared to a strong non-parametric learner such as XGB. Another advantage of LR over tree-based models such as XGB is its ability to extrapolate outside of the feature value ranges observed on accepts [71], which is crucial since rejected applications are coming from a different distribution region.

On each labeling iteration, we randomly sample ρm examples from the available set of m rejects. Sampling aims at preventing overfitting by examining different regions of the distribution of rejects. Assuming that the currently deployed scorecard performs better than random, we expect the *bad* rate in D^r to be higher than that in D^a. To address this, we introduce the imbalance parameter θ. We only label examples in the bottom γ percentile and the top $\gamma\theta$ percentile of the distribution of scores predicted by the weak learner. This ensures that we select rejects with high confidence in the assigned labels and append more *bad* examples than *good* ones by setting $\theta > 1$. The latter helps to increase the *bad* rate in the training sample to approximate the population distribution. The selected labeled rejects are removed from D^r and appended to D^a. After the first iteration, we fix the absolute values of the confidence thresholds and use them on the following iterations.

Training Stage

At the end of each labeling iteration, we train a scoring model on the augmented labeled sample D^a containing accepts and selected labeled rejects. The augmented sample covers a wider range of the feature space compared to the original sample of accepted applications. This helps to reduce the adverse effect of sampling bias on the trained model. The training stage benefits from using a strong base learner to develop a scorecard with high discriminative power to screen new applications. We use XGB as a base classifier for the resulting scorecard.

Early Stopping

The number of labeling iterations is controlled by the stopping criteria. We use the Bayesian evaluation framework proposed in Section 6.4 to track the performance of the corrected scorecard across the labeling iterations. At the end of each iteration, we evaluate the scorecard on a holdout sample containing labeled accepts and unlabeled rejects. Evaluating a model with the Bayesian framework is important as it allows to account for the impact of sampling bias on evaluation. If the model performance does not improve, we stop labeling at this iteration and use the best-performing model as a resulting scorecard. We also specify the maximum number of labeling iterations j_{max} and terminate the BASL algorithm if there are no more rejects in D^r for which predictions exceed the specified confidence thresholds.

6.6 Experimental Setup

This section describes the data used in the paper and outlines the experimental setup. First, we use a controlled simulation environment to illustrate sampling bias, demonstrate gains from our propositions, and investigate boundary conditions affecting their performance. Next, we test our methods and quantify their business impact on a real-world high-dimensional microloan data set.

6.6.1 Synthetic Data

We generate synthetic loan applications using two multivariate mixtures of Gaussian distributions:

$$\begin{cases} \mathbf{X}^g \sim \sum_{c=1}^{C} \delta_c \mathcal{N}_k(\boldsymbol{\mu}_c^g, \boldsymbol{\Sigma}_c^g) \\ \mathbf{X}^b \sim \sum_{c=1}^{C} \delta_c \mathcal{N}_k(\boldsymbol{\mu}_c^b, \boldsymbol{\Sigma}_c^b) \end{cases} \tag{6.6.1}$$

where $\mathbf{X}^g$ and $\mathbf{X}^b$ are feature matrices of *good* and *bad* applications, and δ_c, $\boldsymbol{\mu}_c$, and $\boldsymbol{\Sigma}_c$ are the weight, mean vector and covariance matrix of the c-th mixture component. The distribution parameters control the difference between the two applicant groups.

Mimicking the scorecard-based loan approval process, which leads to sampling bias, we introduce a simulation framework called the acceptance loop. We assume a financial institution approves loan applications using a scoring model $f_a(X)$ that predicts $\mathbf{P}(y = 1|X)$. The institution accepts the applicant X if $f_a(X) \leq \tau$, where τ is a probability threshold. Suppose $D_j = \{(\mathbf{X}, \mathbf{y})\}$ is the batch j of independent and identically distributed applications with $(\mathbf{X}, \mathbf{y}) \sim \mathbf{P}_{XY}$ where $\mathbf{y}$ is unknown at the time of application. Acceptance decisions partition D_j into $D_j^a = \{X_i \in \mathbf{X} | f_a(X_i) \leq \tau\}$ and $D_j^r = \{X_i \in \mathbf{X} | f_a(X_i) > \tau\}$ for accepts and rejects. Once the labels in D_j^a are available, the scoring model is updated by incorporating all labeled applications $D^a = \bigcup_{j=1}^{J} D_j^a$ during training and applied on new incoming applications, where J is the total number of batches. Over time, D^a grows in size with a bias towards accepts.

We run the acceptance loop for 500 iterations. On each iteration, we generate a new batch of applications using the same distribution parameters and train a scoring model $f_a(X)$ over D^a to split them into accepts and rejects. We also draw a representative holdout sample from $\mathbf{P}_{XY}$ denoted as H. The sample H is used to evaluate the performance of scorecards and bias correction methods on unseen data representative of the borrowers' population. A detailed description of the simulation framework and synthetic data generation is provided in Appendix 6.9.3.

Full control over the data generating process facilitates sensitivity analysis to clarify how the loss due to bias and gains from our propositions develop with changes in the environment and uncover boundary conditions. For example, Section 6.2 has discussed missingness mechanisms and how they impact the loss due to bias. Hence, the sensitivity analysis comprises a gradual transition from an MAR to an MNAR process. Other factors influencing the effectiveness of BASL include the strength of the sampling bias, the class imbalance ratio, and the complexity of the classification task. Similarly, the Bayesian framework depends on the validation set of labeled accepts and unlabeled rejects and the quality of the class prior for the labels of rejects. The sensitivity analysis proposes measures for these factors and examines their impact on our propositions.

6.6.2 Real Data

The real-world credit scoring data set is provided by a FinTech called Monedo and constitutes consumer microloans issued to customers in Spain. The data includes 2,410 features characterizing the loan applicants. The target variable is a binary indicator of whether the customer has timely repaid the loan (*good*) or experienced delinquency of at least three consecutive months (*bad*). 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 accepts, whereas the repayment outcome of rejected clients is unknown. Table 6.6.1 summarizes the real-world data set.

Apart from accepts and rejects, we also have access to a labeled unbiased holdout sample with 1,967 customers who have been granted credit without scoring. The sample, therefore, includes examples that would normally be rejected by a scorecard and represents the through-the-door population of customers who apply for a loan. This unbiased sample allows us to evaluate the performance gains from our propositions under the true operating conditions of

Table 6.6.1. Real Data Summary

Characteristic	Accepts	Rejects	Holdout
Number of applications	39,579	18,047	1,967
Number of features	2,410	2,410	2,410
Percentage of *bad* applications	39%	Unknown	66%

Monedo.

Table 6.6.1 shows that the *bad* rate in the holdout sample is 1.7 times higher than among accepts, which hints at the presence of sampling bias. Appendix 6.9.4 provides additional analysis confirming that the data do not exhibit MCAR and illustrating sampling bias and its adverse effect on the scorecard parameters, training and evaluation. The results indicate the potential of bias correction.

6.6.3 Experiments

The empirical evaluation focuses on two research questions. Experiment I tests whether the Bayesian framework provides a more reliable estimate of the scorecard performance on unseen data compared to other evaluation strategies. Experiment II focuses on training under sampling bias and tests whether the BASL framework outperforms conventional bias correction methods.

Experiment I compares evaluation strategies in a performance prediction setup. We split accepts into training and validation sets and apply evaluation strategies to a scorecard trained on the training data. Each strategy provides an estimate of the scorecard performance on a holdout sample representative of the borrowers' population. Ignoring sampling bias and evaluating on accepts is a naive benchmark. DR and reweighting act as off-policy evaluation benchmarks. Differences between the off-policy evaluation setup and our study prohibit the direct application of DR. Appendix 6.9.6 details our implementation of an adjusted DR estimator that supports credit scoring. The Bayesian framework evaluates the scorecard on a merged validation set of accepts and unlabeled rejects. To produce a prior on the labels of rejects, we score them with the XGB-based scorecard trained on accepts and calibrate the scores using LR. We judge the performance of an evaluation strategy by calculating the RMSE between the model performance estimates produced by that strategy over the experimental trials and the actual scorecard performance on the holdout sample.

In Experiment II, we correct the training set of accepts with one of the bias correction methods. The scoring model is trained over the corrected sample and evaluated on a representative holdout sample. We compare BASL to established techniques from different families of bias correction methods. Ignoring rejects serves as a baseline. Labeling rejects as *bad* and bureau score based labeling are simple augmentation techniques popular in credit scoring. HCA and parceling represent the model-based augmentation methods. The Heckman model is another benchmark suited for MNAR and established in the credit scoring literature. We also implement reweighting with cluster-based weights. The bias-removing autoencoder serves as a representation change benchmark.

The simulation study allows us to dynamically conduct the experiments within the acceptance loop and aggregate the results over 100 simulation trials. Knowledge of the actual labels of synthetic rejects also allows us to implement an oracle model $f_o(X)$ trained on

$D^a \cup D^r$. The oracle represents a scorecard that does not suffer from sampling bias and indicates an upper performance bound. The real data is static and does not support dynamic evaluation and an oracle scorecard. To improve the robustness of the results obtained on the real data, we aggregate performance over 100 values coming from 4 cross-validation folds times 25 bootstrap samples of the holdout sample. We use XGB as a base classifier in experiments on both data sets. Further details on the data partitioning and meta-parameter values of bias correction methods are provided in Appendix 6.9.5.

6.7 Results

6.7.1 Synthetic Data

This section presents empirical results on synthetic data. We start by illustrating sampling bias and gains from our propositions in the MAR setup. Next, we perform sensitivity analysis to investigate boundary conditions affecting the performance of BASL and the Bayesian evaluation framework. Last, we compare gains from our propositions and benchmarks depending on the missingness type.

Results in the MAR Setup

Figure 6.7.1 illustrates sampling bias and its adverse effects on the scorecard behavior, training and evaluation. Panel (a) compares the distribution densities of one of the synthetic features in D^a, D^r and H. The results indicate differences in the distribution of x_1 in D^a and H. The values of x_1 in $(-3, 1)$ are not observed among accepts in D^a, although the density peak in the unbiased set H is located within this interval. This confirms that the training data of previously accepted clients are not representative of the population of loan applicants.

Bias in the training data affects the scorecard behavior. We use a non-parametric XGB-based scorecard, which prohibits a direct inspection of the model parameters to illustrate the bias in the classifier. Regressing applicant features on the predictions of an XGB scorecard using linear regression, we obtain a surrogate model that approximates the way in which XGB translates feature values into predictions. Panel (b) compares the coefficients of the surrogate models corresponding to the three XGB scorecards: (i) biased model $f_a(X)$ trained over D^a; (ii) oracle model $f_o(X)$ trained over $D^a \cup D^r$; (iii) model $f_c(X)$ corrected by labeling rejects with BASL. The results indicate that sampling bias affects the coefficients of surrogate scorecards and causes them to diverge from the oracle values. BASL partly recovers this difference, bringing the coefficients closer to the oracle. The bias in model parameters translates into a difference in the scores predicted by the scorecards. As illustrated in panel (c), f_a provides more optimistic scores compared to f_o, whereas the distribution of scores produced by f_c is more in line with that of the unbiased model.

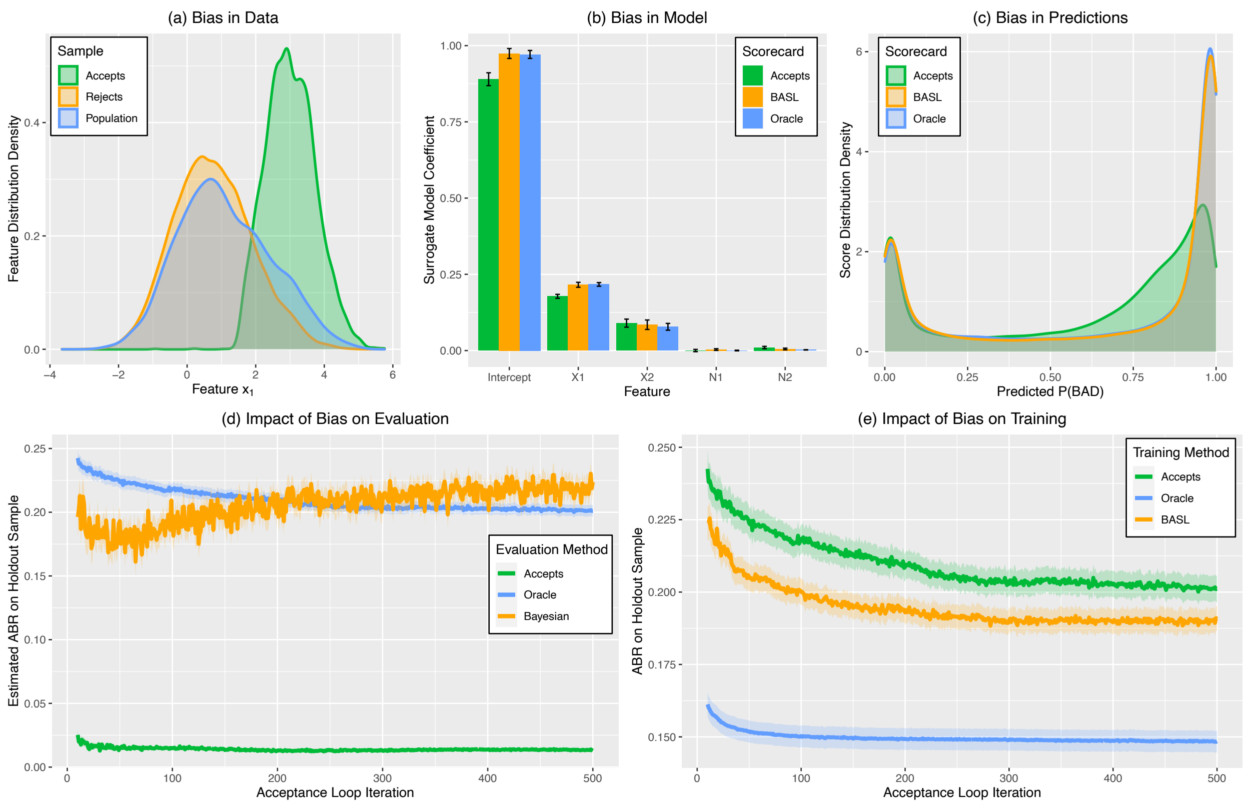

Figure 6.7.1. Loss due to Sampling Bias and Gains from Our Propositions

The figure illustrates sampling bias, its adverse effect on the scorecard behavior, training and evaluation, and gains from our propositions within the acceptance loop on synthetic data. Panel (a) demonstrates bias in the distribution of one of the features. Panel (b) visualizes bias in scorecards by comparing the coefficients of the corresponding linear surrogate models. Panel (c) shows bias in scorecard predictions for new loan applications in the holdout sample. Panels (d) and (e) depict the impact of bias on scorecard training and evaluation in terms of the ABR. The meta-parameters of the data generation process and the bias correction methods are provided in Appendix 6.9.5.

Panel (d) of Figure 6.7.1 depicts the results of Experiment I that analyzes the impact of sampling bias on the scorecard evaluation. It compares the ABR of f_a on the representative holdout sample H (labeled as oracle performance) and the estimated ABR of f_a using accepts-based evaluation or Bayesian evaluation. Evaluating f_a on a sample from D^a provides an overoptimistic estimate of around .0140. However, when applied to new applications from H, the scorecard demonstrates the ABR of around .2095. This gap illustrates that the accepts-based performance estimate is misleading due to sampling bias. The Bayesian framework provides a more reliable estimate of the ABR, reducing the mean RMSE between the actual and predicted ABR values from .2010 to .0929.

Panel (e) illustrates the results of Experiment II, depicting the effect of sampling bias on the scorecard performance. It compares the ABR of f_a, f_o and f_c. The performance of f_a gradually improves over acceptance loop iterations due to the increasing training sample size. The ABR of f_o remains stable after 100 iterations and is consistently lower than that of f_a. The difference between f_o and f_a captures the loss due to sampling bias. The average

ABR loss across 100 simulation trials is .0598. The model corrected with BASL consistently outperforms f_a starting from the first iteration of the acceptance loop. Note that f_c still suffers from the bias and does not reach the ABR of f_o. However, BASL consistently recovers about 25% of the ABR loss compared to f_a.

Similar performance gains from BASL and the Bayesian framework in Experiment I and II are observed in other evaluation metrics, the AUC, PAUC and BS, and reported in Appendix 6.9.3. Following the recommendations of Demšar [29], we conclude that the observed gains are statistically significant. Friedman's rank sum tests reject the null hypothesis that our propositions perform the same as ignoring rejects in both experiments (p-values are lower than 2.2×10^{-16}). Pairwise Nemenyi post-hoc tests indicate that our propositions significantly outperform the accept-based training and evaluation at a 5% significance level for each of the four metrics.

Sensitivity Analysis

This section examines factors that determine the effectiveness of our propositions to understand when their use is recommended and identify boundary conditions.

Figure 6.7.2 depicts loss due to bias and gains from BASL across different simulations. Panel (a) examines the influence of the magnitude of sampling bias in the training data. The bias magnitude is controlled by the acceptance rate, which varies across financial products and markets. Lower acceptance results in a greater difference between D^a and D. This raises the loss due to sampling bias and the effectiveness of BASL to mitigate this loss. We find BASL to improve scorecards for acceptance rates below 20%, whereby this result originates from assuming a 30% *good* rate in the borrowers' population. Increasing the *good* rate would lead to a slower decay of the loss due to bias and performance gains from BASL. Still, the analysis implies that BASL is more helpful for financial products with low acceptance rates,

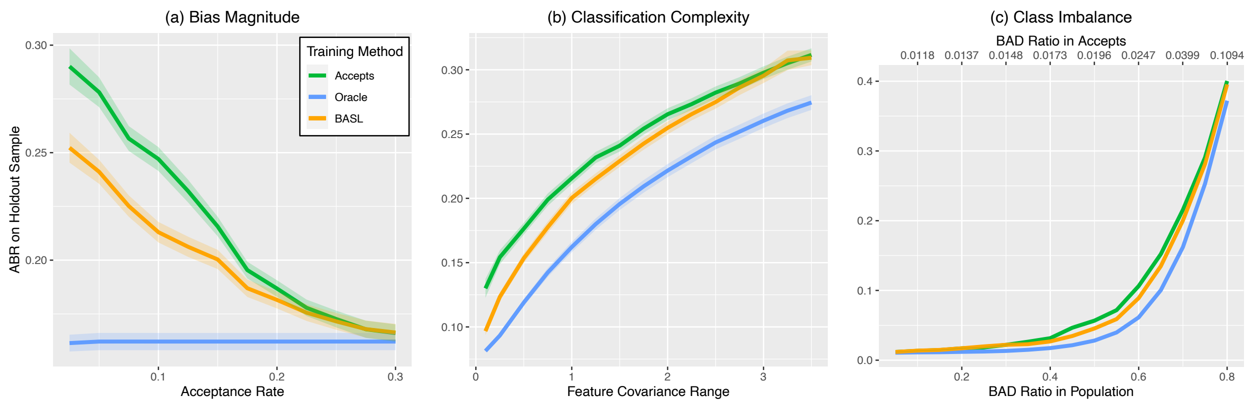

Figure 6.7.2. Sensitivity Analysis: Bias-Aware Self-Learning

The figure illustrates the impact of sampling bias on the model training and benefits from reject inference with BASL as a function of three parameters: (i) bias magnitude controlled by the acceptance rate; (ii) classification task complexity controlled by the feature covariance range; (iii) class imbalance controlled by the *bad* rate in population.

such as, e.g., installment loans for prime customers. The results also agree with Crook and Banasik [26], who find a negative relationship between the acceptance rate and performance gains from reweighting-based bias correction.

Panel (b) studies the classification complexity and depicts the development of scorecard performance as a function of the feature covariance range. The elements of the feature covariance matrix are drawn randomly. A wider range of possible covariance values increases the classification complexity because loan applications of different classes tend to overlap more frequently in the feature space. The loss due to sampling bias is consistently present across the considered complexity range. At the same time, performance gains from BASL are higher in environments with a lower classification complexity and gradually diminish in more complex environments. This is explained by the fact that the pseudo-labels assigned to rejects are more accurate when class separation is easier. The ability to distinguish *good* and *bad* applicants is, therefore, an important factor affecting the potential usefulness of reject inference. In practice, observed default rates can shed light on the complexity of the classification task associated with scoring applications for a financial product.

Panel (c) investigates the impact of class imbalance, which we control by the proportion of *bad* applications in the population. The results suggest that any *bad* rate in the population translates into imbalance among accepts since the data is filtered by a scorecard. The loss due to bias shrinks when class imbalance becomes too strong. This is observed because the ABR metric only focuses on the least risky applicants, which are mostly *good* due to high imbalance. BASL provides the largest gains at moderate imbalance between 2% and 5% among accepts. This imbalance level is sufficiently high so that an accepts-based model is not exposed to enough *bad* risks but is not too severe to prohibit learning from the scarce number of *bad* applications.

Turning attention to the Bayesian evaluation framework, panel (a) of Figure 6.7.3 examines the effect of the acceptance rate on scorecard evaluation. To isolate this effect, we assume a perfect prior when calculating the Bayesian extension of the ABR. Under this assumption, the Bayesian framework estimates scorecard performance accurately across all acceptance rates. Similar to BASL, potential gains from Bayesian evaluation are higher at lower acceptance, as the inconsistency between the performance on accepts versus that on a representative sample becomes stronger.

Calculating the Bayesian extension requires a validation sample of labeled accepts and unlabeled rejects. Panel (b) studies how the quality of this sample affects evaluation. We assess sample quality using the maximum mean discrepancy metric [MMD, 15], which measures the similarity of the feature distribution in the validation set and the unbiased holdout set. The results reinforce accept-based evaluation to underestimate error rates substantially. To predict scorecard performance accurately, the Bayesian framework requires validation data that matches the target distribution in the holdout set. To ensure this, the validation sample should include accepts and rejects from the same time period and match the accept/reject

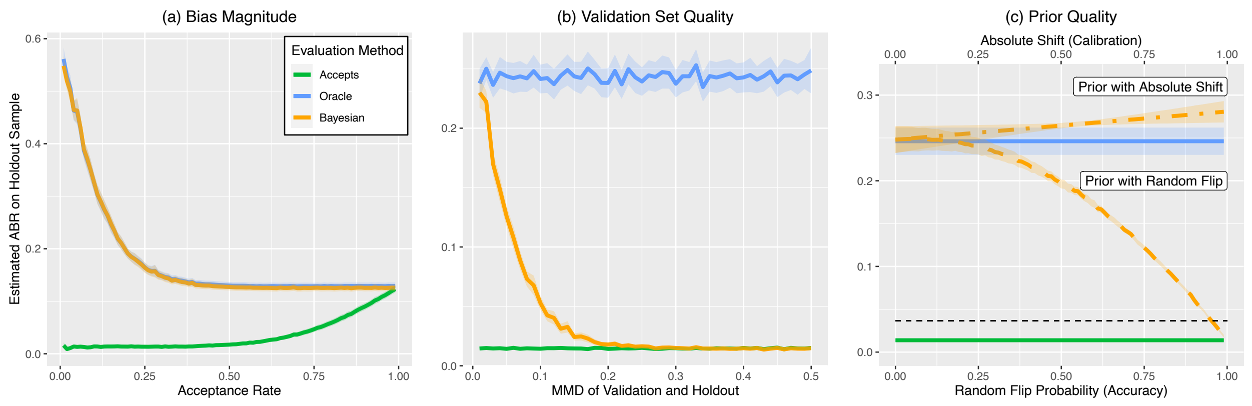

Figure 6.7.3. Sensitivity Analysis: Bayesian Evaluation

The figure illustrates the impact of sampling bias on scorecard evaluation and benefits from the Bayesian framework as a function of three parameters: (i) bias magnitude controlled by the acceptance rate; (ii) validation set quality controlled by the accept/reject rate and measured by the MMD metric; (iii) prior quality controlled by injecting noise in form of a random label flip or level shift. A dashed black line on panel (c) indicates the ABR predicted by the Bayesian framework when using empirical *bad* rate on accepts as a prior for labeling rejects.

ratio in the holdout sample.

Panel (c) focuses on a key ingredient of the Bayesian framework, the class prior for the labels of rejects. Reducing the quality of the estimate of the prior decreases the accuracy of Bayesian evaluation. The quality reduction can stem from diluting the accuracy or the calibration of the prior, which we simulate by injecting random flips or level shifts. The Bayesian framework provides more accurate performance estimates than accepts-based evaluation even when the prior contains a lot of noise. This implies that using imperfect scores (e.g., from a currently used scorecard) as a prior for the labels of rejects still produces a better result than ignoring rejects during evaluation.

Missingness Type

We obtain the previous results on the data generated according to an MAR process, where all applicants' features are revealed to the scorecard. Next, we introduce MNAR by generating data with three explanatory features and using only the first two in the scorecard, creating the omitted variable bias. On each iteration of the acceptance loop, we overwrite a certain percentage of the scorecard-based acceptance decisions by replacing applicants with the lowest values of the hidden feature. This allows us to gradually increase the strength of the MNAR process in the synthetic data. Figure 6.7.4 illustrates the loss due to sampling bias and the performance of our propositions depending on the overwriting percentage. We also implement two established bias correction benchmarks. BASL is compared to the Heckman model, which is developed for MNAR settings, whereas the Bayesian framework is compared to the DR evaluation.

We observe adverse effects of sampling bias on the training and evaluation of scorecards at any level of overwriting. This confirms that bias correction is required regardless of whether the data exhibits MAR or MNAR. In both extremes with no or full manual overwriting, loss in the ABR exceeds .05 during training. The impact on evaluation is even stronger, with a mean difference of .21 between the ABR estimated on accepts and the actual ABR on holdout applications.

Our propositions consistently outperform accepts-based training and evaluation. Panel (a) of Figure 6.7.4 reveals the poor performance of the Heckman model under MAR. We explain this result by a high correlation between the two variables considered in the model: outcome (i.e., whether the applicant is *bad*) and selection (i.e., whether the applicant was accepted). Previous studies also noted a poor Heckman performance in the presence of strong target-missingness correlation [21]. In our context, the correlation is high when scorecards are accurate. Traditional banks that focus on prime customers often obtain low default rates, which indicates the high accuracy of their scorecards. Increasing the overwriting percentage reduces the target-missingness correlation and strengthens the magnitude of the MNAR process, facilitating a superior performance of the Heckman model. In our simulations, Heckman outperforms BASL once more than 20% of the applications undergo overwriting. Such a large degree of manual intervention is not typical for financial institutions that increasingly rely on the automation of approval decisions. More generally, since the type of missingness is difficult to identify in practice, consistent superiority of BASL over the accepts-based benchmark suggests that institutions adopting BASL will not run the risk of impeding scorecard performance; which they might when using the Heckman model.

Considering model evaluation, we observe the accuracy of the ABR estimates from the Bayesian framework to decrease in the MNAR setting. However, they are much closer to the true scorecard performance than estimates coming from accepts-based evaluation and DR across all setups. DR improves over ignoring rejects but still provides ABR estimates that are, on average, more than .10 off the actual model performance. This large gap can

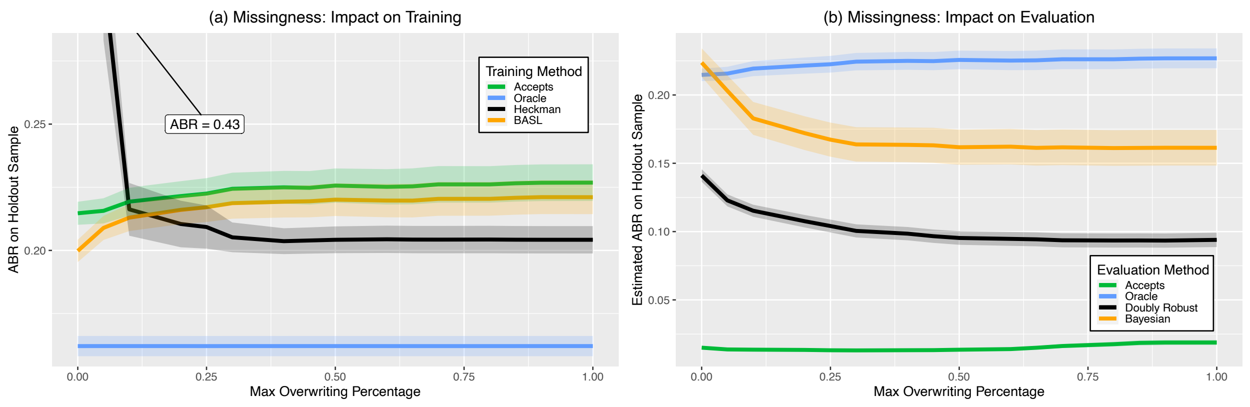

Figure 6.7.4. Sensitivity Analysis: Missingness Type

The figure illustrates the relationship between overwriting and the impact of sampling bias on scorecard training and evaluation, benefits from our propositions (BASL and Bayesian evaluation) and benchmarks (Heckman model and doubly robust evaluation). The vertical axis of panel (a) is truncated at ABR = 0.40 for visualization purposes.

be explained by the adjustments required to apply DR in credit scoring, which we discuss in Appendix 6.9.6.

6.7.2 Real Data

This section presents the results of Experiment I and II on the real-world data set and reports on a business impact analysis, in which we examine the monetary gains from our propositions.

Experiment I: Evaluation with the Bayesian Framework

Table 6.7.1 compares the accuracy of different model evaluation strategies. We compute the RMSE between the scorecard performance estimates produced by each evaluation strategy and the actual scorecard performance on the holdout sample representing the true borrower population.

In line with previous results from synthetic data, we observe relatively high RMSE values when ignoring rejects, which evidences the loss due to sampling bias. Overoptimistic estimates of scorecard performance from the accepts-based evaluation lead to sub-optimal decisions and fail to capture the scorecard's business value. Expecting a certain default rate upon scorecard deployment, a financial institution would face losses and potential liquidity problems when encountering a substantially higher default rate on new loan applications.

Weighted validation improves the accuracy of the scorecard performance estimates for two evaluation metrics. Overall, reweighting performs marginally worse than accepts-based evaluation, achieving an average rank of 2.49 compared to 2.46. At the same time, reweighting outperforms accepts-based evaluation in the metrics that account for asymmetric error costs, the PAUC and the ABR, which are of high importance for decision-makers. DR demonstrates a poor RMSE for the two supported metrics, the BS and ABR. This can be attributed to the high difficulty of reward prediction in a high-dimensional environment and the limitations of DR when applied to credit scoring. Poor performance in the BS and ABR

Table 6.7.1. Scorecard Evaluation: Performance of Bias Correction Methods

Evaluation method	AUC	BS	PAUC	ABR	Rank
Ignore rejects	.1234 (.0309)	.0306 (.0034)	.0983 (.0246)	.0356 (.0603)	2.46
Reweighting	.1277 (.0601)	.0348 (.0054)	.0826 (.3058)	.0315 (.0903)	2.49
Doubly robust	–	.0506 (.0050)	–	.1167 (.0216)	–
Bayesian evaluation	**.0111** (.1158)	**.0073** (.0213)	**.0351** (.0628)	**.0130** (.0331)	**1.06**

Abbreviations: AUC = area under the ROC curve, BS = Brier Score, PAUC = partial AUC on FNR $\in [0, .2]$, ABR = average *bad* rate at 20-40% acceptance, rank = average rank across the four performance measures. Values indicate RMSE between the actual scorecard performance on the holdout sample and performance estimates obtained with a given evaluation method. Variance of the performance estimates $\times 10^{-5}$ in parentheses.

while lacking support for rank-based indicators such as the AUC and PAUC make DR an inappropriate evaluation method for the considered data set.

The Bayesian evaluation framework provides the most accurate estimates of the scorecard performance across all evaluation metrics and achieves an average rank of 1.06. This implies that Bayesian evaluation produces the most reliable predictions of scorecard performance on new loan applications, helping decision-makers to anticipate the accuracy of a scorecard and judge its value ex ante. Appendix 6.9.4 augments Table 6.7.1 with results from statistical testing. Pairwise Nemenyi post-hoc tests indicate that performance estimates obtained with the Bayesian framework are significantly better than those obtained with the benchmark strategies at a 5% level.

Experiment II: Reject Inference with BASL

Table 6.7.2 compares the performance of bias correction methods. We find some methods to perform worse than disregarding rejects. Only three approaches have a lower rank than ignoring rejects. Labeling rejects as *bad* performs worst. Given a historical acceptance rate of $20-40\%$ at Monedo, the underlying assumption of all rejects being *bad* risks is too strong for the used data set. The bias-removing autoencoder also performs poorly. As discussed in Appendix 6.9.6, due to a large number of features and a broad set of meta-parameters, the reconstruction error of the autoencoder remains high even after much tuning. This evidences the difficulty of using an autoencoder in high-dimensional settings.

The Heckman model improves on the previous benchmarks but performs worse than ignoring rejects. Hence, relying on this approach is also ineffective for the data considered here. The poor performance of Heckman can be attributed to two reasons. First, a parametric Heckman model faces difficulties in handling high-dimensional and noisy data. To address this, we consider model variants with reduced feature subsets. The best-performing variant

Table 6.7.2. Scorecard Training: Performance of Bias Correction Methods

Training method	AUC	BS	PAUC	ABR	Rank
Ignore rejects	.7984 (.0010)	.1819 (.0004)	.6919 (.0010)	.2388 (.0019)	3.58
Label all as bad	.6676 (.0014)	.2347 (.0006)	.6384 (.0010)	.3141 (.0022)	8.56
Bias-removing autoencoder	.7304 (.0011)	.2161 (.0004)	.6376 (.0019)	.3061 (.0036)	7.79
Heckman model	.7444 (.0011)	.2124 (.0006)	.6397 (.0010)	.3018 (.0013)	7.53
Bureau score based labels	.7978 (.0009)	.1860 (.0003)	.6783 (.0010)	.2514 (.0021)	4.97
Hard cutoff augmentation	.8033 (.0010)	.1830 (.0006)	.6790 (.0011)	.2458 (.0021)	4.25
Reweighting	.8040 (.0005)	.1840 (.0002)	.6961 (.0009)	.2346 (.0015)	3.45
Parceling	.8038 (.0011)	.1804 (.0004)	.6885 (.0011)	.2396 (.0019)	3.32
Bias-aware self-learning	**.8166** (.0007)	**.1761** (.0003)	**.7075** (.0011)	**.2211** (.0012)	**1.55**

Abbreviations: AUC = area under the ROC curve, BS = Brier Score, PAUC = partial AUC on FNR $\in [0, .2]$, ABR = average *bad* rate at 20-40% acceptance, rank = average rank across the four measures. Standard errors in parentheses.

presented in Table 6.7.2 includes the 65 most important features, which we selected using permutation-based importance. Second, in line with the synthetic data results, the Heckman model performs poorly when the outcome and selection equations are highly correlated. The correlation increases with the accuracy of the previous scorecard. High correlation is also more typical for data exhibiting MAR. Although it is not feasible to reliably estimate the strength of the MNAR process on the real data, the poor performance of Heckman could imply that the missingness type is more geared towards MAR.

Considering established model-based augmentation techniques, HCA improves on ignoring rejects only in the AUC, whereas parceling performs better in two evaluation measures. The better performance of parceling can be explained by introducing randomness on the labeling stage, which helps this approach reduce the error propagation and achieve an overall rank of 3.32.

Reweighting outperforms other benchmarks in the AUC, PAUC and ABR. Despite the good performance in these measures, reweighting has a worse BS than ignoring rejects, indicating a poor calibration ability of the resulting scorecard. This translates to a marginally higher overall rank of reweighting compared to parceling. Appendix 6.9.6 discusses the performance of different reweighting variants in more detail, whereas Table 6.7.2 only includes the best-performing specification.

BASL performs the best in each performance indicator and achieves the lowest average rank of 1.55. Compared to reweighting, the closest competitor in the cost-sensitive metrics, the PAUC and ABR of the scorecard after bias correction with BASL increase by .0114 and .0135, respectively. Gains from BASL are statistically significant: Nemenyi post-hoc tests indicate that BASL significantly outperforms all benchmarks at a 5% level in the AUC, PAUC, and ABR. Appendix 6.9.4 provides auxiliary results from an ablation study, which examines incremental performance gains from different stages of BASL. The largest gains are attributed to the filtering stage.

Business Impact Analysis

To evaluate the business impact of our propositions, we estimate gains in monetary performance. This requires knowledge of key loan parameters. We consider two financial products with different properties: microloans and installment loans. Microloans are small-amount, short-term, often single-payment, high-interest loans. Installment loans have larger amounts, smaller interest, and are repaid over time by regular payments. The approval rates for microloans tend to be higher than those for installment loans.

We draw loan amounts and interest rates from Gaussian distributions with means set to the values observed in the US consumer loan market[12]. We compute i as the total interest and fees divided by the principal A. The loss given default (LGD) indicates the percentage

[2]Source: The Pew Charitable Trusts (2016) Payday Loan Facts, `https://www.pewtrusts.org/-/media/assets/2016/06/payday_loan_facts_and_the_cfpbs_impact.pdf`.

Table 6.7.3. Business Settings

Parameter	Notation	(a) Microloans	(b) Installment loans
Acceptance rate	α	$[.2, .4]$	$[.1, .2]$
Loan principal	A	\$375 (SD = \$100)	\$17,100 (SD = \$1,000)
Total interest rate	i	.1733 (SD = .01)	.1036 (SD = .01)
Loss given default	LGD	$[0, 1]$	$[0, 1]$

The table reports parameters of the business impact analysis. Principals and interest rates are drawn from Gaussians with reported means and standard deviations (in parentheses). The LGD is drawn from [0, 1] with a step of .01.

of A lost in case of default and varies between 0 and 1. Table 6.7.3 provides the parameter values for the two markets.

In the event of default occurring with a probability PD, a financial institution recovers $A \times (1+i) \times (1-\text{LGD})$. If there is no default, the expected revenue is $A \times (1+i)$. For each bias correction method, we approximate the loan-level PD by computing the ABR of this method within the specified acceptance range. We use the modeling pipeline of Section 6.6 to obtain 100 ABR estimates for each bias correction method. Given these 100 estimates and the values from Table 6.7.3, Equation 6.7.2 yields an estimate of the average profit per loan for every bias correction method:

$$\pi = \frac{1}{100}\sum_{j=1}^{100}\left[\text{PD}_j \times A \times (1+i) \times (1-\text{LGD}) + (1-\text{PD}_j) \times A \times (1+i) - A\right] \qquad (6.7.2)$$

We aggregate the average profit per loan over 10,000 trials, drawing A and i from the Gaussian distributions and varying the LGD from 0 to 1. By subtracting the profit of each bias correction method from the profit of a scorecard that ignores rejects, we compute the incremental profit compared to ignoring sampling bias. Finally, we compute the expected margin (i.e., the expected return per dollar issued) by dividing the incremental profit by the average loan amount. It is worth noting that the expected profit assumes that all applications are either *good* or *bad*. In reality, more outcomes are possible: e.g., customers can repay early or consolidate into a different loan.

Figure 6.7.5 illustrates the expected return as a function of the LGD. We focus on the two bias correction methods achieving the lowest ABR: BASL tuned with the Bayesian evaluation framework and reweighting. Ignoring sampling bias impacts the profit of a financial institution. On the microloan market, BASL increases the expected return per dollar issued by up to 2.07 percentage points compared to ignoring rejects and up to 1.58 percentage points compared to the best benchmark. For installment loans, monetary gains are up to 2.70 percentage points compared to ignoring rejects and 2.18 compared to reweighting. Assuming the loan amounts reported in Table 6.7.3, the incremental profit from correcting

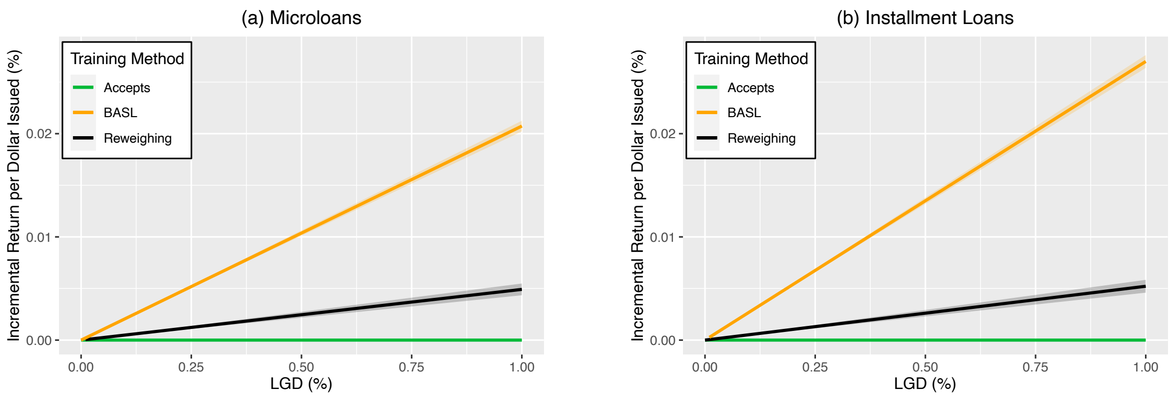

Figure 6.7.5. Monetary Gains

The figure illustrates the expected incremental return per dollar issued of BASL and reweighting relative to a scorecard that ignores sampling bias on two markets: (a) microloans and (b) installment loans.

sampling bias with our propositions is up to $7.78 per loan on the microloan market and up to $461.70 per loan on the market of installment loans.

6.8 Conclusion

Scoring models aid decision-making by predicting the outcomes of choice alternatives. Studying loan approval decisions in retail finance, the paper illustrates the adverse effects of sampling bias on training and evaluation of scoring models and addresses both with two propositions. The Bayesian evaluation framework leverages unbiased unlabeled data for estimating model performance reliably. The BASL framework mitigates the loss due to bias by training scorecards on a debiased sample, which comprises selected rejects with an inferred label. Using real-world lending data including an unbiased sample representative of the true borrower distribution, we confirm the effectiveness of BASL and the Bayesian framework, demonstrate their superiority over other bias correction methods, and find reject inference to increase profit. Results from a simulation study augment those findings by identifying boundary conditions that determine the effectiveness of our propositions.

The study has several implications for credit scoring. Regulatory frameworks such as the Basel Accord require banks to establish internal capital adequacy assessment procedures, which involve the validation of scorecards. We show that widely used evaluation practices are vulnerable to sampling bias and give misleading advice. The Bayesian framework anticipates the loss in accuracy due to bias and facilitates quantifying this effect prior to scorecard deployment. A more realistic and forward-looking performance evaluation helps to improve risk management practices in the industry and benefits the empirical credit scoring literature, which routinely assesses scoring models using biased data of accepted clients [e.g., 57].

The paper also shows that training on biased data decreases the accuracy and profitability of a scorecard. Financial institutions can use BASL in combination with any supervised learning algorithm to reduce the loss in model performance. Doubt as to whether reject inference is worthwhile prevails in the literature [e.g., 21]. Reporting positive results from an unbiased evaluation sample, the paper speaks to this scepticism. Reject inference is a hard problem. Financial rewards will not be excessive. However, the specific engineering of BASL facilitates consistent and material gains in this study. Improvements of the magnitude observed here in a core business process may well be a deciding factor in highly competitive lending markets.

Exploiting the potential of reject inference and our propositions requires access to unbiased unlabeled data. Meeting this requirement in a credit context is nontrivial. Financial institutions need to store data on rejected applicants, which poses challenging questions related to privacy and consumer protection. Balancing the interests of lenders to gather more data for improving processes such as loan approval and the interests of consumers for protection against privacy infringement is a major challenge in the digital economy. Quantifying the value of a specific type of data in a specific business use case, the paper contributes a humble piece of empirical evidence to this societal debate, which may inform bank governance and regulatory authorities.

The increasing use of scoring models to derive predictions and recommendations from observational data in various fields warrants general concern about sampling bias. The growing literature on off-policy evaluation and learning echoes these concerns and provides approaches for the robust evaluation and learning of policies in a contextual bandit setup. To our best knowledge, corresponding methods have received minimal attention in credit scoring, where the outcomes or rewards associated with a reject decision are never observed. Based on a simulation study and experiments on real-world lending data, we find that BASL and the Bayesian framework outperform selected off-policy benchmarks. These results are specific to our data and experimental design, which reflect the characteristics of a credit scoring context. Hence, they evidence that our propositions deserve a place in data scientists' toolbox and can offer superior decision support in certain scenarios.

Performing sensitivity analysis and examining boundary conditions, the paper offers several criteria to anticipate the loss due to sampling bias in an application setting and the suitability of the proposed remedies. We find that the magnitude of the loss due to bias and the potential recovery from bias correction is higher in environments with low acceptance rates, moderate or high class imbalance and good class separation. Class separability is dependent on the available features and difficult to measure in real life. Class imbalance, on the other hand, is a known modeling challenge encountered in many scoring model applications [e.g., 88]. The last characteristic, termed low approval rate in a credit context, refers to the amount of labeled data that is available for model training and evaluation. Applications in which the acquisition of labels is costly or involves the allocation of a scarce resource display this characteristic.

The characteristics indicate when the loss due to bias is likely substantial. How to address sampling bias is a different question. One way to mitigate bias involves gathering a representative evaluation and/or training sample by experimentation. Bias correction methods such as BASL and the Bayesian framework should be considered whenever a random allocation of resources is very costly, prohibited, or unethical, which can be the case in medical applications. A criterion to judge the suitability of the Bayesian framework is the observability of decision outcomes (or policy rewards). In credit scoring, the repayment status of a loan is observable only if the application was accepted. Off-policy evaluation methods require adjustments to support this peculiarity, which complicates their use and may harm their effectiveness. Hence, the Bayesian framework is especially suitable in scenarios where certain actions do not reveal rewards. The same consideration applies when measuring scoring model performance using indicators like the AUC, which cannot be calculated on the level of an individual case. For BASL, we observe relative advantages over alternatives like the Heckman model if the process that governs the relationship between outcomes and features and the labeling process (i.e., selection equation) are strongly correlated. Finally, the sensitivity analysis emphasizes the generality of the problem by confirming that sampling bias diminishes the accuracy of scoring model performance estimates independent of whether class labels are missing at random (MAR) or not at random (MNAR). Concerning model training, the status-quo in the credit scoring literature suggests that scorecards lose accuracy under MNAR, whereas posterior probability models like logistic regression do not require debiasing under MAR [8]. Our analysis extends this result by showing that tree-based models, which fail to extrapolate outside of the observed feature ranges, benefit from bias correction even in the MAR setting.

The discussion of environmental characteristics offers guidance when to worry about sampling bias and helps to identify scenarios that could benefit from our propositions. Consider the example of fraud detection, which involves processing a vast amount of transactions or insurance claims and generating model-based fraud scores. Pointing analysts to the most suspicious cases, the scores facilitate efficient utilization of fraud screening resources. Fraud labels are known for an often small subset of previously investigated cases, and the share of fraudulent cases is very low [97]. These characteristics mimic the low acceptance and high imbalance setting in our simulation and suggest that sampling bias might be a serious issue. Given an abundance of unlabeled data and noting that fraud labels (outcomes) remain unknown unless investigating a case, BASL and the Bayesian framework may have the potential to enhance fraud detection practices.

Other interesting examples come from medical settings. Being well aware of the risks of sampling bias, randomized trials and off-policy learning and evaluation are well established in the field. Exemplary use cases of scoring models include treatment allocation decisions. Outcomes relate to recipients' health or well-being and these can be observed independent from taking a specific action (e.g., do not depend on prescribing a treatment). However, scoring models also inform the allocation of transplants to patients on a waiting list by predicting, e.g., post-transplant survival [17]. Here, an outcome is observed for the low percentage of candidates previously selected for transplant but never observed when rejecting a recipient. This causes class imbalance, creates sampling bias, and mimics the scenario studied in the paper, which proved challenging for off-policy evaluation methods. Gathering representative data through experiments is also not an option. Thus, the validation of the scores is a major problem in transplant allocation, which the Bayesian framework could address.

The examples underline the generality of the sampling bias problem and the vast space of applications for debiasing techniques in management and beyond. They also illustrate how use cases of scoring models in different fields share characteristics of the credit scoring context studied in this paper. Ignoring sampling bias affects the efficiency of resource allocation decisions and may have adverse implications for the people affected by those decisions. The two contributions proposed in the paper constitute a holistic approach to sampling bias mitigation and can be used together or on a standalone basis to raise decision quality and create value.

6.9 Appendix

6.9.1 Prior Work on Bias Correction

This appendix includes the literature tables that provide a comprehensive overview of bias correction methods suggested in different research streams and summarize previous empirical studies on reject inference in credit scoring. A detailed description of the prominent bias correction methods is provided in Section 6.3.

Bias Correction Methods

Table 6.9.1 overviews sampling bias correction methods suggested in the prior work. The bias correction methods suggested in different research streams are grouped into three families depending on the application stage: data preprocessing, model training and model evaluation. Data preprocessing methods encompass representation change techniques that transform input data before modeling to reduce the bias between the source and target distributions. Methods that correct sampling bias in the training stage split into two subgroups: model-based and reweighting techniques. Model-based techniques are embedded in a learning algorithm and account for the bias during model training by adjusting the optimization problem. Reweighting methods rebalance the loss function of a learning algorithm towards more representative data examples and can be applied during model training and model evaluation. Apart from reweighting, methods that correct bias in the evaluation stage include multiple evaluation metrics that approximate the generalization error under sampling bias and metric-agnostic evaluation frameworks.

Table 6.9.1. Sampling Bias Correction Methods

Reference	Method	Type	DP	TR	EV	MA	NT
Blitzer et al. [14]	Structural correspondence learning	RC	✓			✓	
Daumé III [27]	Supervised feature augmentation	RC	✓			✓	
Saenko et al. [83]	Supervised feature transformation	RC	✓			✓	
Gopalan et al. [38]	Sampling geodesic flow	RC	✓			✓	
Gong et al. [37]	Geodesic kernel flow	RC	✓			✓	
Caseiro et al. [20]	Unsupervised feature transformation	RC	✓			✓	
Saptal et al. [84]	Penalized feature selection	RC	✓			✓	
Pan et al. [80]	Transfer component analysis	RC	✓			✓	
Long et al. [68]	Transfer joint matching	RC	✓			✓	
Sun et al. [95]	Correlation alignment	RC	✓			✓	
Wang et al. [100]	Extreme dimension reduction	RC	✓			✓	
Atan et al. [3]	Bias-removing autoencoder	RC	✓			✓	
Heckman [42]	Heckman's model	MB		✓			✓
Meng et al. [77]	Heckman-style bivariate probit	MB		✓			✓
Lin et al. [60]	Modified SVM	MB		✓			✓
Daumé III et al. [28]	Maximum entropy genre adaptation	MB		✓			✓
Yang et al. [105]	Adapt-SVM	MB		✓			✓
Marlin et al. [73]	Multinomial mixture model	MB		✓			✓
Bickel et al. [13]	Kernel logistic regression	MB		✓			✓
Chen et al. [23]	Co-training for domain adaptation	MB, RC	✓	✓		✓	
Duan et al. [30]	Domain adaptation machine	MB		✓			✓
Long et al. [67]	Regularized least squares	MB		✓			✓
Liu et al. [64]	Robust bias-aware classifier	MB		✓			✓
Joachims et al. [48]	Modified ranking SVM	MB		✓			✓
Chen et al. [24]	Robust bias-aware regression	MB		✓			✓
Liu et al. [63]	Modified bias-aware classifier	MB		✓			✓
Kügelgen et al. [56]	Semi-generative model	MB		✓			✓
Rosenbaum et al. [81]	Model-based probabilities	RW		✓	✓	✓	✓
Shimodaira [86]	Distribution density ratios	RW		✓	✓	✓	✓
Zadrozny [106]	Selection probabilities are known	RW		✓	✓	✓	✓
Huang et al. [45]	Kernel mean matching	RW		✓	✓	✓	✓
Cortes et al. [25]	Cluster-based frequencies	RW		✓	✓	✓	✓
Sugiyama et al. [93]	Kullback-Leibler weights	RW		✓	✓	✓	✓
Kanamori et al. [50]	Least-squares importance fitting	RW		✓	✓	✓	✓
Loog [69]	Nearest-neighbor based weights	RW		✓	✓	✓	✓
Gong et al. [36]	Focusing on cases similar to test data	RW		✓	✓	✓	✓
Shimodaira [86]	Modified AIC	EM			✓		✓
Sugiyama et al. [94]	Subspace information criterion	EM			✓		✓
Sugiyama et al. [92]	Generalization error	EM			✓		✓
Sugiyama et al. [91]	Importance-weighted validation	EF			✓	✓	✓
Bruzzone et al. [18]	Circular evaluation strategy	EF			✓	✓	✓
This paper	**BASL and Bayesian evaluation**	**MB, EF**		✓	✓	✓	✓

Method types: RC = representation change, MB = model-based, RW = reweighting, EM = evaluation metric, EF = evaluation framework. Applications stages: DP = preprocessing, TR = training, EV = evaluation. Other abbreviations: MA = model-agnostic method, NT = does not involve input data transformation.

Table 6.9.2. Empirical Studies on Reject Inference in Credit Scoring

Reference	Implemented technique(s)	Training method(s)	Evaluation method(s)	Representative holdout sample	Profit gains	No. features
Joanes [49]	Reclassification	DA	–			3
Fogarty [33]	Multiple imputation	DA	–			10
Xia [103]	Outlier detection with isolation forest	DA	–			9
Liu et al. [66]	Ensembling classifiers and clusteres	MB	–			5, 23
Kang et al. [51]	Label spreading with oversampling	DA	–			22
Boyes et al. [16]	Heckman model variant (HM)	MB	–			42
Feelders [32]	Mixture modeling	MB	–			2
Chen et al. [21]	HM	MB	–	✓		24
Banasik et al. [9]	HM	MB	–	✓		30
Wu et al. [102]	HM	MB	–			2
Kim et al. [54]	HM	MB	–	✓		16
Chen et al. [22]	Bayesian model	MB	–		✓	40
Li et al. [59]	Semi-supervised SVM (S3VM)	MB	–			7
Marshall et al. [75]	HM	MB	–			18
Tian et al. [96]	Kernel-free fuzzy SVM	MB	–			7, 14
Xia et al. [104]	CPLE-LightGBM	MB	–			5, 17
Anderson [1]	Bayesian network	MB	–			7, 20
Kim et al. [53]	S3VM with label propagation	MB	–			17
Shen et al. [85]	Unsupervised transfer learning	MB	–			20
Banasik et al. [7]	Banded weights	RW	–	✓		30
Verstraeten et al. [98]	Resampling	RW	–	✓	✓	45
Bücker et al. [19]	Missing data based weights	RW	–			40
Crook et al. [26]	Banded weights, extrapolation	RW, DA	–	✓		30
Banasik et al. [8]	HM with banded weights	MB, RW	–	✓		30
Maldonado et al. [70]	Self-learning, S3VM	MB, DA	–			2, 20, 21
Anderson et al. [2]	HCA, Mixture modeling	DA, MB	–			12
Nguyen [78]	Parceling, HM, Banded weights	DA, MB, RW	–			9
Mancisidor et al. [72]	Bayesian model, self-learning, S3VM	DA, MB	–			7, 58
This paper	**BASL, Bayesian evaluation**	**DA**	**EF**	✓	✓	**2,410**

Abbreviations: DA = data augmentation, MB = model-based, RW = reweighting, EF = evaluation framework. “Representative holdout sample” indicates whether the study has access to a sample from the borrower’s population for evaluation.“Profit gains” indicates whether gains from bias correction are measured in terms of profit. “Heckman model variant” includes a linear Heckman model and a bivariate probit/logistic model with non-random sample selection.

In addition to the method type and the application stage, Table 6.9.1 indicates two further characteristics of the bias correction methods: (i) whether the method is model-agnostic and (ii) whether it requires input data transformation. The advantage of model-agnostic methods is their flexibility with respect to the base classifier. Methods that rely on input data transformation require training a scoring model on latent features, which may harm the comprehensibility and explainability of the model.

Applications in Credit Scoring

Table 6.9.2 overviews empirical studies on bias correction in credit scoring. We group the studies by the type of the implemented bias correction technique(s) and distinguish the methods applied in the model training and model evaluation stage. The discussion of the findings in Table 6.9.2 is available in Section 6.3.3.

The empirical studies on reject inference are summarized across multiple dimensions, including: (i) whether the employed data set includes a holdout sample representative of the population, (ii) whether the performance gains are measured in profit and (iii) the number of features in the data set. The first dimension illustrates the potential for an accurate estimation of gains from bias correction, which requires labeled applications rejected by a scorecard. The second dimension shows if improvements from bias correction are measured in terms of the monetary gains. The third dimension indicates the data set dimensionality. The importance of handling the high-dimensional feature spaces is rising in light of a growing market share of FinTechs that operate with large amounts of data from different sources [89] and an increasing reliance of financial institutions on alternative data sources such as applicant's digital footprint, e-mail activity and others [e.g. 47].

6.9.2 Bias-Aware Self-Learning Framework

This appendix consists of two parts: (i) it provides the pseudo-code describing the BASL reject inference framework; (ii) it elaborates on the benefits of using a weak learner such as LR to label rejected applications.

Framework Pseudo-Code

BASL includes four stages: (i) filtering rejects, (ii) labeling rejects, (iii) training the scorecard, (iv) early stopping. Algorithm 4 provides the pseudo-code describing the filtering stage. Algorithm 5 describes the labeling stage. The complete BASL framework is summarized in Algorithm 6 and explained in Section 6.5.

Labeling Rejects with a Weak Learner

This section illustrates the importance of using a weak learner on the labeling stage of the BASL framework. Consider two scoring models that employ different base classifiers: (i)

```
input  : accepts $\mathbf{X}^a$, rejects $\mathbf{X}^r$, meta-parameters $\beta = (\beta_l, \beta_u)$
output: filtered rejects $\mathbf{X}^r$
1 $g(X)$ = novelty detection algorithm trained over $\mathbf{X}^a$ ;
2 $\mathbf{s}^r = g(\mathbf{X}^r)$ ;                    // predict similarity scores
3 $\mathbf{X}^r = \{X_i \in \mathbf{X}^r | s^r_i \in [\beta_l, \beta_u]\ \}$, $\beta_l$ and $\beta_u$ are percentiles of $\mathbf{s}^r$ ;      // filter
   rejects
return: $\mathbf{X}^r$
```

Algorithm 4: Bias-Aware Self-Learning: Filtering Stage

```
input  : labeled accepts $D^a = \{(\mathbf{X}^a, \mathbf{y}^a)\}$, unlabeled rejects $D^r = \{\mathbf{X}^r\}$,
         meta-parameters $\rho, \gamma, \theta$
output: selected labeled rejects $D^* = \{(\mathbf{X}^*, \mathbf{y}^*)\}$
1 $\mathbf{X}^* = \text{sample}(\mathbf{X}^r, \rho)$, $\rho$ is the sampling rate ;     // random sample of rejects
2 $f(X)$ = weak learner trained over $\mathbf{D}^a$ ;
3 $\mathbf{s}^* = f(\mathbf{X}^*)$ ;                    // score rejects with a weak learner
4 derive $c_g$: $\mathbf{P}(\mathbf{s}^* < c_g) = \gamma$, $\gamma$ is the percentile threshold ;
5 derive $c_b$: $\mathbf{P}(\mathbf{s}^* > c_b) = \gamma\theta$, $\theta$ is the imbalance parameter ;
6 $\mathbf{X}^* = \{X^*_i \in \mathbf{X}^* | s^*_i < c_g \text{ or } s^*_i > c_b\}$ ;       // select relevant examples
7 $\mathbf{y}^* : y^*_i = 0$ if $s^*_i < c_g$ and $y^*_i = 1$ if $s^*_i > c_b$ ;          // assign labels
return: $\{(\mathbf{X}^*, \mathbf{y}^*)\}$
```

Algorithm 5: Bias-Aware Self-Learning: Labeling Stage

```
input  : labeled accepts $D^a = \{(\mathbf{X}^a, \mathbf{y}^a)\}$, unlabeled rejects $D^r = \{\mathbf{X}^r\}$, holdout
         set $H = \{\mathbf{X}^h\}$; meta-parameters: filtering ($\beta$), labeling ($\tau, \gamma, \theta$), stopping
         criteria ($j_{max}$, $\mathbf{P}(y^r|\mathbf{X}^r), \epsilon$)
output: corrected scoring model $f_c(X)$
1  $\mathbf{X}^r = \text{filtering}(\mathbf{X}^a, \mathbf{X}^r, \beta)$ ;                  // filtering stage
2  $j = 0$; $V = \{\}$; $F = \{\}$ ;                          // initialization
3  while ($j \leq j_{max}$) and ($\mathbf{X}^r \neq \emptyset$) and ($V_j \geq V_{j-1}$) do
4      $j = j + 1$
5      $D^* = \{(\mathbf{X}^*, \mathbf{y}^*)\} = \text{labeling}(D^a, D^r, \rho, \gamma, \theta)$ ;          // labeling stage
6      $D^a = D^a \cup D^*$ ;      // append labeled rejects to the labeled sample
7      $\mathbf{X}^r = \mathbf{X}^r - \mathbf{X}^*$ ; // remove labeled rejects from the unlabeled sample
8      $F_j = f(X)$ = strong learner trained over augmented $D^a$ ; // training stage
9      $V_j = \text{BM}(F_j, H, \mathbf{P}(y^r|X^r), \epsilon)$ ;      // evaluate using a Bayesian metric
10 end
11 return $f_c(X) = F_{\arg\max(V)}$ ;      // return best-performing strong learner
```

Algorithm 6: Bias-Aware Self-Learning Framework

$f_{XGB}(X)$ uses a strong non-parametric learner (XGB); (ii) $f_{LR}(X)$ uses a weak learner (L1-regularized LR). Using a real-world data set described in Section 6.6, we train both models on D^a and use them to score applications in the holdout sample H.

Figure 6.9.1 compares the distribution densities of predictions of f_{XGB} and f_{LR}. LR produces probabilistic predictions that follow a distribution with a high kurtosis and a density peak around .40. In contrast, XGB scores follow a distribution with heavier tails. The range of the scores of f_{XGB} is wider than that of f_{LR}, and the predictions are close to extreme values of 0 or 1 more frequently. This indicates a poor calibration of boosting predictions, which is also confirmed by the previous studies [79].

Within the BASL framework, we only label rejects for which the labeling model has high confidence in the predicted labels, focusing on the examples from the tails of the score distribution. This requires a labeling algorithm that is able to accurately assign labels for examples on the extremes of the score distribution and has a good calibration ability to facilitate setting the appropriate confidence thresholds based on the predicted probabilities of default. As indicated in Figure 6.9.1, a weak learner such as LR is a good fit. A weak learner is also less prone to overfitting the biased training sample, which could increase the risk of error propagation during labeling. Furthermore, as explained in Section 6.5, a regression-type algorithm such as LR is able to extrapolate outside of the feature range observed in the training data, which is not the case for tree-based learners such as XGB. Therefore, we use LR on the labeling stage of BASL.

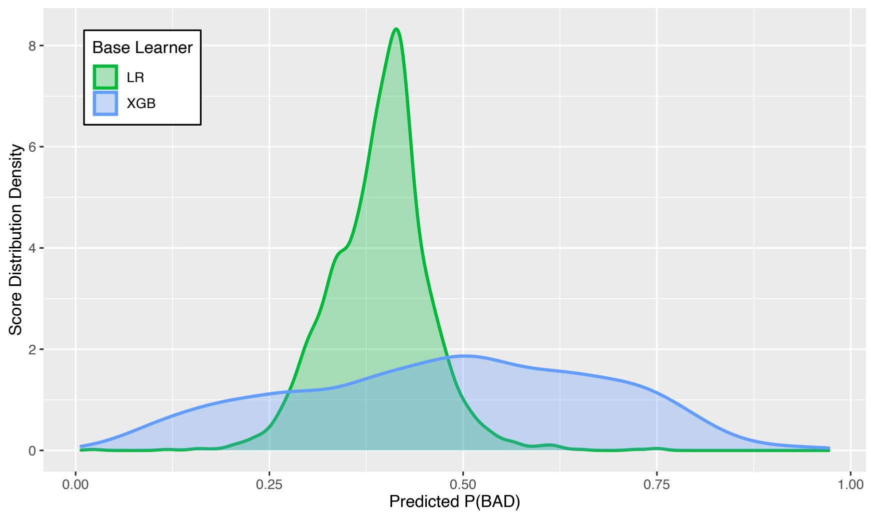

Figure 6.9.1. Prediction Density Comparison

The figure compares distribution densities of the scores predicted by two scoring models using different base classifiers: (i) L1-regularized logistic regression f_{LR} (green); (ii) extreme gradient boosting f_{XGB} (blue).

6.9.3 Extended Results on Synthetic Data

This appendix provides methodological details of the simulation framework introduced in Section 6.4 and additional empirical results that extend the results reported in Section 6.7.1. The simulation study illustrates sampling bias, its adverse effect on the behavior, training and evaluation of scoring models and gains from our two propositions: the Bayesian evaluation framework and BASL. The parameters of the data generation process and the acceptance loop used in the simulation are provided in Appendix 6.9.5.

Simulation Framework

The simulation framework is summarized in Algorithm 8. The framework consists of two stages: the initialization and the acceptance loop. In the initialization stage, we generate synthetic data including two classes of borrowers from a mixture of Gaussian distributions using Algorithm 7. A similar approach to generate synthetic loan applications using Gaussian distributions has been used in the prior work [e.g., 76, 70]. Let our synthetic examples $\mathbf{X}^g = (X_1^g, ..., X_n^g)^\top$ and $\mathbf{X}^b = (X_1^b, ..., X_m^b)^\top$ representing *good* and *bad* loan applications be generated as follows:

$$\begin{cases} \mathbf{X}^g \sim \sum_{c=1}^{C} \delta_c \mathcal{N}_k(\boldsymbol{\mu}_c^g, \boldsymbol{\Sigma}_c^g) \\ \mathbf{X}^b \sim \sum_{c=1}^{C} \delta_c \mathcal{N}_k(\boldsymbol{\mu}_c^b, \boldsymbol{\Sigma}_c^b) \end{cases} \tag{6.9.3}$$

where δ_c is the weight of the c-th Gaussian function, $\sum_{c=1}^{C} \delta_c = 1$, and $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$ are the mean vector and the covariance matrix of the c-th Gaussian. The elements of $\boldsymbol{\Sigma}_c^i$ are drawn from a uniform distribution $\mathcal{U}(0, 1)$. We also append two noisy features with the same mean and variance for both classes: $x_\varepsilon \sim \mathcal{N}(0, 1)$.

Suppose the random binary vector $\mathbf{y} = \mathbf{y}^g \cup \mathbf{y}^b$ is a label indicating if an applicant is a *good* ($y = 0$) or *bad* risk ($y = 1$). The difference between the applicant classes is controlled by the parameters of the underlying distributions. Assuming a *bad* rate of b, we generate $n_b = nb$ *bad* examples and $n_g = n(1 - b)$ *good* examples and construct a first batch of the loan applications $D^* = \{(\mathbf{X}^*, \mathbf{y}^*)\}$ with $(\mathbf{X}^*, \mathbf{y}^*) \sim \mathbf{P}_{XY}$. We also generate a holdout set of h examples denoted as $H = \{(\mathbf{X}^h, \mathbf{y}^h)\}$ using the same parameters as for the initial population. H acts as a representative set that does not suffer from sampling bias. We use H for performance evaluation.

The second stage of the framework – the acceptance loop – simulates the dynamic acceptance process, where loan applications arrive in batches over certain periods of time (e.g., every working day). Assume that $D^* = \{\mathbf{X}^*\}$ is the first batch of n applicants a financial institution encounters when entering a new market. Since no repayment data have been collected so far, a company might rely on a simple business rule to filter applications. An example would be to rank applications in D^* by their credit bureau scores denoted as x_v. In our simulation, x_v refers to a feature with the largest difference in mean values between *good* and *bad* applicants and represents a powerful attribute, such as a bureau score, that can be used to perform a rule-based application ranking. Assuming the target acceptance rate of α, the financial institution grants a loan to αn applicants with the highest bureau scores, forming a set of accepts $D^a = \{X_i \in \mathbf{X}^* | x_{i,v} \geq \tau\}$, and reject $(1-\alpha)n$ remaining applicants, forming a set of rejects $D^r = \{X_i \in \mathbf{X}^* | x_{i,v} \leq \tau\}$, where τ is the $(1-\alpha)$-th percentile of x_v with respect to D^*. Eventually, the repayment status of applicants in D^a is observed, providing the corresponding labels $\mathbf{y}^a$. The labeled set $D^a = \{(\mathbf{X}^a, \mathbf{y}^a)\}$ can now be used to train a scoring model $f_a(X)$ to support the acceptance decisions for the incoming loan applications.

On each iteration of the acceptance loop, $f_a(X)$ is trained over the available set of accepts in D^a. In addition to $f_a(X)$, we also train an oracle model $f_o(X)$ over the union of accepts

```
input  : distribution parameters $\boldsymbol{\mu}_c^g, \boldsymbol{\mu}_c^b, \boldsymbol{\Sigma}_c^g, \boldsymbol{\Sigma}_c^b, \delta_c, C$, sample size $n$, *bad* ratio $b$
output: labeled set of examples $D = \{(\mathbf{X}, \mathbf{y})\}$
1 $n_b = bn$; $n_g = n - n_b$ ;                          // compute class-specific sample sizes
2 $\mathbf{X}^g \sim \sum_{c=1}^{C} \delta_c \mathcal{N}_k(\boldsymbol{\mu}_c^g, \boldsymbol{\Sigma}_c^g)$ ;        // generate $n_g$ good applications
3 $\mathbf{X}^b \sim \sum_{c=1}^{C} \delta_c \mathcal{N}_k(\boldsymbol{\mu}_c^b, \boldsymbol{\Sigma}_c^b)$ ;         // generate $n_b$ bad applications
4 $\mathbf{y}^g = \vec{0}$; $\mathbf{y}^b = \vec{1}$ ;                        // define applications' labels
5 $D = \{(\mathbf{X}^g, \mathbf{y}^g) \cup (\mathbf{X}^b, \mathbf{y}^b)\}$ ;                   // construct a data set
return : $D$
```

Algorithm 7: Synthetic Data Generation

```
input  : distribution parameters $\boldsymbol{\mu}_c^g, \boldsymbol{\mu}_c^b, \boldsymbol{\Sigma}_c^g, \boldsymbol{\Sigma}_c^b, \delta_c, C$, sample sizes $n, h$, *bad* ratio
         $b$, acceptance rate $\alpha$, number of iterations $j_{max}$, feature indicator $v$
output: labeled accepts $D^a$, labeled rejects $D^r$, labeled holdout set $H$
1  $D^* = \{(\mathbf{X}^*, \mathbf{y}^*)\} = \text{generate}(\boldsymbol{\mu}_c^g, \boldsymbol{\mu}_c^b, \boldsymbol{\Sigma}_c^g, \boldsymbol{\Sigma}_c^b, \delta_c, C, b, n)$ ; // generate data using
    Algorithm C.1
2  $H = \{(\mathbf{X}^s, \mathbf{y}^s)\} = \text{generate}(\boldsymbol{\mu}_c^g, \boldsymbol{\mu}_c^b, \boldsymbol{\Sigma}_c^g, \boldsymbol{\Sigma}_c^b, \delta_c, C, b, h)$ ; // generate holdout set
3  $\tau = (1-\alpha)$-th percentile of $x_v$ with respect to $D^*$ ;   // simple business rule
4  $D^a = \{(X_i^*, y_i^*) | x_{i,v} \geq \tau\}$ ;                  // accept $\alpha n$ applications
5  $D^r = \{(X_i^*, y_i^*) | x_{i,v} < \tau\}$ ;              // reject $(1-\alpha)n$ applications
6  for $j \in \{1, 2, ..., j_{max}\}$ do
7      $f_a(X)$ = accepts-based model trained over $D^a$ ;
8      $f_o(X)$ = oracle model trained over $D^a \cup D^r$ ;
9      $D_j = \{(\mathbf{X}, \mathbf{y})\} = \text{generate}(\boldsymbol{\mu}_c^g, \boldsymbol{\mu}_c^b, \boldsymbol{\Sigma}_c^g, \boldsymbol{\Sigma}_c^b, \delta_c, C, b, n)$ ;        // batch of new
        applications
10     $\tau = \alpha$-th percentile of $f_a(D_j)$ ;          // compute acceptance threshold
11     $D_j^a = \{(X_i, y_i) | f_a(X_i) \leq \tau\}$ ;              // accept $\alpha n$ applications
12     $D_j^r = \{(X_i, y_i) | f_a(X_i) > \tau\}$ ;          // reject $(1-\alpha)n$ applications
13     $D^a = \bigcup_{i=1}^{j} D_i^a$; $D^r = \bigcup_{i=1}^{j} D_i^r$ ;            // append accepts and rejects
14 end
return : $D^a, D^r, H$
```

Algorithm 8: Simulation Framework

and rejects $D^a \cup D^r$. The model f_o represents an upper performance bound as it uses representative data that is not available in practice and does not suffer from sampling bias. We use both f_a and f_o to score examples in H and evaluate their performance. Next, we generate a batch of n new applicants $D_j = \{(\mathbf{X}, \mathbf{y})\}$ using the same distribution parameters as for the initial population (i.e., assuming the absence of population drift) and predict the scores of the new applicants in using f_a. Based on the model predictions, we accept αn examples with the lowest predicted scores and reject the remaining $(1 - \alpha)n$ applications. The newly rejected examples are appended to D^r, whereas the newly accepted examples with their labels are appended to D^a. The augmented set of accepts is used to retrain f_a on the next iteration of the acceptance loop.

To illustrate performance gains from BASL, we apply it on each iteration of the acceptance loop. We train a scoring model $f_c(X)$ that has undergone bias correction by applying BASL to the available set of rejects D^r and augmenting the training set D^a with the selected labeled rejects. On each iteration, we train f_c on the augmented data and score examples in H. This allows us to track gains from BASL compared to f_a.

Gains from the Bayesian evaluation framework are demonstrated by comparing the actual performance of f_a on a representative holdout sample H (labeled as oracle performance) and the predicted performance of f_a estimated with different evaluation methods on each iteration of the acceptance loop. First, f_a is evaluated on a validation subset drawn from the available set of accepts D^a. Second, we evaluate f_a on a validation set that consists of the labeled accepts in D^a and pseudo-labeled rejects in D^r using the Bayesian evaluation framework. This allows us to quantify the gap between the actual and predicted performance of the scorecard and measure the recovery of this gap by using the Bayesian framework for performance evaluation.

Experiment I

This section provides the extended results of Experiment I on synthetic data in the MAR setup. Table 6.9.3 compares the performance of accepts-based evaluation and the Bayesian evaluation framework. The table quantifies the difference between the actual scorecard performance on a representative holdout set and the predicted scorecard performance estimated with one of the two evaluation strategies. We measure bias, variance and RMSE of the performance estimates using four evaluation metrics: the AUC, BS, PAUC and ABR. The results are aggregated across 100 simulation trials $\times$ 500 acceptance loop iterations.

According to Table 6.9.3, performance estimates provided by the Bayesian evaluation framework have a lower bias than those obtained within the accepts-based evaluation. Despite accepts-based estimates demonstrating a lower variance in two evaluation metrics, the BS and ABR, RMSE values between the actual and predicted scorecard performance clearly indicate the advantage of using the Bayesian framework for scorecard evaluation. In all considered evaluation metrics, the Bayesian framework is able to provide a better estimate

Table 6.9.3. Experiment I Results on Synthetic Data

Evaluation method	Metric	Bias	Variance	RMSE
Accepts-based evaluation	AUC	.1923	.0461	.2205
Bayesian framework		.0910	.0001	.1000
Accepts-based evaluation	BS	.0748	.0006	.0828
Bayesian framework		.0038	.0009	.0566
Accepts-based evaluation	PAUC	.2683	.0401	.2803
Bayesian framework		.1102	.0002	.1187
Accepts-based evaluation	ABR	.1956	.0004	.2010
Bayesian framework		.0039	.0040	.0929

Abbreviations: AUC = area under the ROC curve, BS = Brier Score, PAUC = partial AUC on FNR $\in [0, .2]$, ABR = average *bad* rate among accepts at 20-40% acceptance, RMSE = root mean squared error.

of the scorecard performance on unseen cases from the borrowers' population.

Experiment II

Table 6.9.4 presents the extended results of Experiment II on synthetic data in the MAR setup. The table provides the average loss due to sampling bias using five metrics. First, we use four scorecard performance metrics, the AUC, BS, PAUC and ABR, to measure the performance deterioration. The loss due to bias is measured as a difference between the performance of the oracle model trained on the union of accepts and rejects and that of the accepts-based model trained on accepts only. Second, we measure the loss in the MMD metric, which represents the magnitude of sampling bias in the labeled training data. The MMD is calculated between the training data of accepts and the representative holdout sample. The gains from reject inference with BASL are measured as a percentage from the corresponding loss due to sampling bias in each metric. The results are averaged across 100 simulation trials $\times$ 500 acceptance loop iterations.

The results suggest that the loss due to sampling bias is observed in all considered performance metrics. BASL consistently recovers between 22% and 36% of the loss. The largest performance gains are observed in the AUC and the BS, which represent the metrics that disregard error costs and are measured on the full set of credit applicants. The gains in the two cost-sensitive metrics measured on the subset of applications deemed as least risky, the PAUC and the ABR, are smaller but still exceed 22%. This suggests that gains from reject inference are observed through both type I and type II error reduction.

Interestingly, the results in the MMD metric indicate that augmenting the training data of accepts with rejects labeled by BASL improves the MMD by just 3.74%. This implies that the training data still exhibits a strong sampling bias. At the same time, using that data to train a corrected scoring model recovers more than 22% of the loss due to bias and

Table 6.9.4. Experiment II Results on Synthetic Data

Metric	Loss due to sampling bias	Gain from BASL
AUC	.0591	35.72%
BS	.0432	29.29%
PAUC	.0535	22.42%
ABR	.0598	24.82%
MMD	.5737	3.74%

Abbreviations: AUC = area under the ROC curve, BS = Brier Score, PAUC = partial AUC on FNR $\in [0, .2]$, ABR = average *bad* rate among accepts at 20-40% acceptance, MMD = maximum mean discrepancy.

scorecard performance, respectively. This result emphasizes that it is enough to label only a portion of rejected cases that help to improve the predictive performance and is further supported by the results of the accuracy-bias trade-off analysis provided in Appendix 14. Increasing the number of labeled rejects allows to further improve the MMD, but does not lead to better scorecard performance due to the noise in the assigned labels. The trade-off between introducing too much noise in the labels and gains from having more representative training data is, therefore, a crucial part of BASL.

Bias-Accuracy Trade-Off

This section investigates the trade-off between sampling bias in the data used for scorecard development and scorecard accuracy. Using synthetic data, we compare multiple variants of BASL and reject inference techniques that perform data augmentation (i.e., label rejected applications and append them to the training data). The results demonstrate the importance of limiting the number of labeled rejects to obtain the best performance and illustrate the relationship between performance maximization and bias mitigation.

The analysis is performed on the synthetic data, which we describe in detail in Appendix 6.9.3. After running the acceptance loop, we apply different reject inference techniques to augment the biased training data of accepts and measure the accuracy and bias of each technique. First, we evaluate the performance of each reject inference method using the four performance metrics considered in the paper: the area under the ROC curve (AUC), the partial AUC (PAUC), the Brier score (BS) and the average *bad* rate among accepts (ABR). Second, we evaluate the magnitude of sampling bias in the augmented training data after reject inference. Here, we use the maximum mean discrepancy metric [MMD, 15], which measures the feature distribution similarity between the augmented training data and the holdout sample.

We implement different variants of BASL, varying the meta-parameter values such that a different subset of rejects is selected during the labeling iterations of the framework. This allows us to consider BASL variants that arrive at different points in the accuracy-bias space. Labeling more rejects facilitates reducing sampling bias, as the distribution mismatch between the training data and the holdout sample diminishes when adding more rejected applications. On the other hand, noise in the pseudo-labels assigned to the appended rejects harms the performance of the resulting scorecard. To study the trade-off between these conflicting dimensions, we construct Pareto frontiers that contain the non-dominated BASL solutions in the bias-accuracy space.

It is important to emphasize that our approach to measuring bias after reject inference is only suitable for data augmentation methods that label rejects and expand the training data. Some reject inference methods (e.g., the Heckman model) do not explicitly label rejects. Therefore, apart from BASL, our experiment includes four data augmentation benchmarks: ignoring rejects, labeling all rejects as *bad* risks, hard cutoff augmentation (HCA) and parceling. In addition to the Pareto frontiers with non-dominated BASL variants, we also depict some dominated BASL solutions with a high MMD to sketch the bias-accuracy relationship when labeling fewer rejects. For that purpose, we split the MMD interval between the non-dominated BASL variant with the highest MMD and ignoring rejects into equal bins and display the best-performing BASL variant within each of the bins. Figure 6.9.2 demonstrates the results.

As depicted in Figure 6.9.2, ignoring rejects leads to the strongest sampling bias in the training data since it only includes accepts, exhibiting MMD of around .60. The data augmentation benchmarks – labeling rejects as *bad*, HCA and parceling – completely eliminate sampling bias and reduce the MMD to around 0. Such a low MMD is achieved by labeling all rejects, which provides training data that represents the borrower population. However, a high reduction in the MMD does not necessarily improve the performance of the corrected scorecard. Except for the AUC, where all benchmarks improve on ignoring rejects, only some of the three data augmentation techniques outperform the scorecard that ignores rejects. This can be explained by the noise in the pseudo-labels of rejects that results from labeling all rejects, including those that are very different from the accepts.

The BASL framework includes multiple steps to restrict the labeling to selected rejects and attend to the distribution similarity between accepts and rejects and the model's confidence in the assigned label. Limiting the number of labeled rejects substantially decreases the gain in MMD. The BASL variants lying on the Pareto frontiers label between 3% and 42% of the rejects after multiple labeling iterations. This allows decreasing the MMD to some value in the range between .40 and .15, indicating that the training data still exhibits sampling bias. We obtain the best performance from scorecards that make use of only a small part of the labeled rejects (around 3% for the BS and 9% for the other evaluation metrics). The best dominated BASL variants lying outside of the Pareto frontiers further reduce the number of labeled rejects to between 1% and 3%. This harms the performance compared to the best solutions on the frontiers but still allows outperforming the considered data augmentation benchmarks.

Overall, the results indicate that there is a trade-off between reducing sampling bias and improving scorecard performance. This trade-off depends on the quality of the labels assigned to the rejected applications. Naturally, correctly labeling all rejects and appending them to the training data would maximize both the performance and the distribution similarly. In practice, predicted labels of rejects are noisy, which makes labeling too many rejects harm scorecard performance. At the same time, labeling too few rejects does not allow to fully realize the potential of reject inference, as demonstrated by the performance of the dominated BASL scorecards. This bias-accuracy relationship forces a decision-maker to settle for a trade-off. In our paper, we focus on the model accuracy as the ultimate

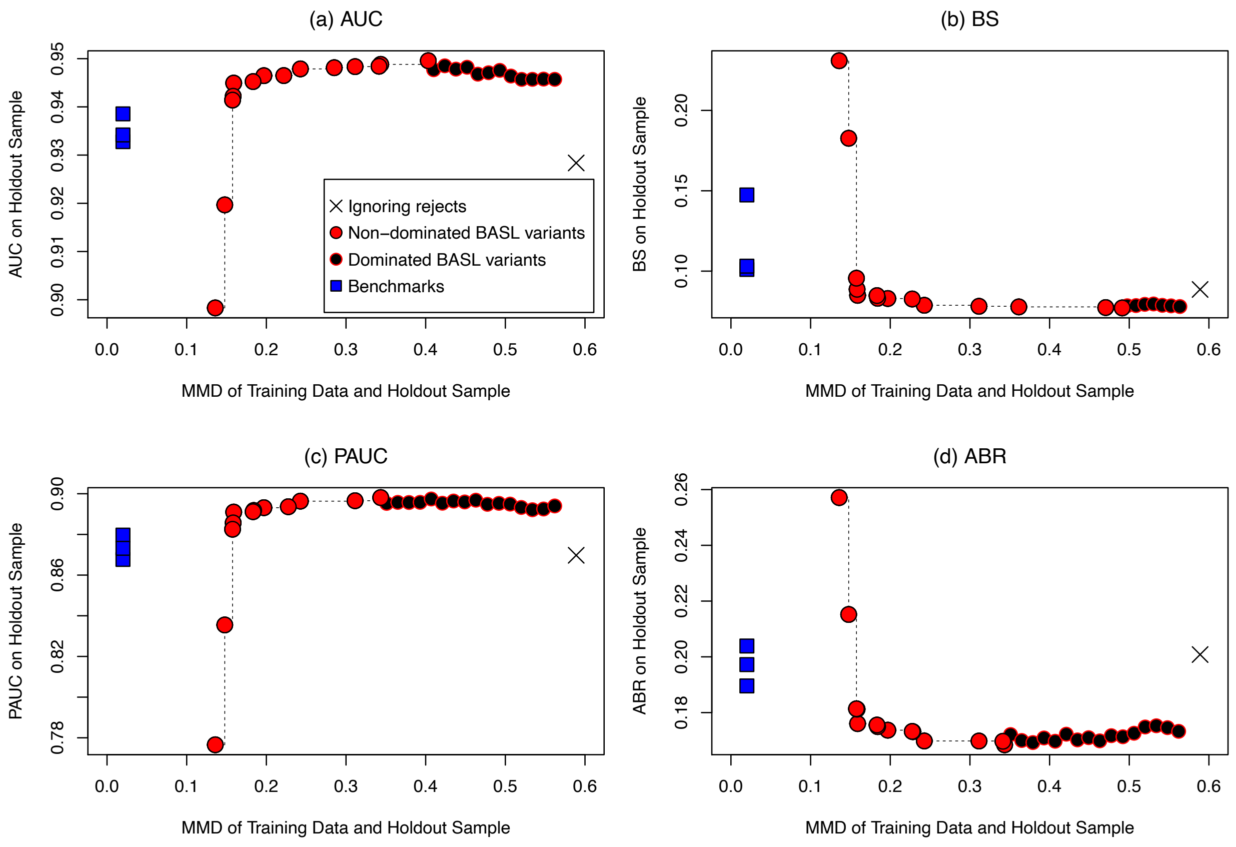

Figure 6.9.2. Bias-Accuracy Trade-Off of Reject Inference Techniques

The figure illustrates the trade-off between the scorecard accuracy and sampling bias when performing reject inference with data augmentation techniques. The vertical axes measure the scorecard performance in one of the four metrics: area under the ROC curve (AUC), Brier score (BS), partial AUC on FNR $\in [0, .2]$ (PAUC), average *bad* rate among accepts at 20-40% acceptance (ABR). The horizontal axes quantify sampling bias in the (augmented) training data by calculating the mean maximum discrepancy (MMD) between the training data and the representative holdout sample. The black cross refers to ignoring rejects. The blue squares depict data augmentation benchmarks that label all rejects: label all as *bad*, hard cutoff augmentation and parceling. The red points depict non-dominated BASL variants with different meta-parameter values. The non-dominated BASL variants label between 3% and 42% of rejects. The black points refer to the best dominated BASL variants that label between 1% and 3% of rejects.

goal of bias correction and tune the meta-parameters of BASL to optimize the scorecard performance.

6.9.4 Extended Results on Real Data

This appendix provides additional empirical results on the real-world credit scoring data set. First, we demonstrate the presence of sampling bias in the sample of accepted applications and illustrate its impact on scorecard behavior, training and evaluation. Second, we provide extended results of Experiment I and II.

Sampling Bias Illustration

Due to the high dimensionality of the real-world data, we focus on a subset of important features to illustrate sampling bias. First, we remove features that exhibit high multicollinearity by applying correlation-based filtering (Spearman or Pearson correlation above .95), which reduces the number of features from 2,410 to 1,549. Second, we train an XGB-based scorecard to produce estimates of the feature permutation importance. We rank features by their importance and use the most important features in the following analysis.

Figure 6.9.3 illustrates sampling bias and its adverse effects on credit scorecards. Panel (a) compares the distribution densities of the most important feature denoted as x_1 on D^a, D^r and H. The results show clear differences in the feature distribution. This observation is supported by the results of three statistical tests over the top-ten most important features at the 1% significance level. Little's mean-based test rejects the null hypothesis that the labels are missing completely at random, indicating the presence of sampling bias [61]. The Probit-based Lagrange Multiplier test reaches the same conclusion and rejects the null hypothesis of the absence of non-random sample selection [74]. Finally, the kernel MMD test indicates the difference in the feature distribution between the accepts and the holdout sample [39]. Overall, the results indicate that the data exhibits sampling bias in previously accepted applications.

Bias in the training data affects the scorecard behavior. Panel (b) compares the coefficients of the top-five most important features of two exemplary scorecards: (i) biased model $f_a(X)$ trained over D^a; (ii) oracle model $f_o(X)$ trained over H. Both scorecards use LR as a base classifier. The results indicate that sampling bias in accepts affects coefficients of the trained scorecard, causing them to diverge from the oracle values. The differences are observed in coefficient sizes (e.g., for x_1) as well as in their signs (e.g., for x_4). The bias in the model parameters translates into a difference in the scores predicted by the scorecards illustrated in panel (c). The accepts-based model f_a provides more optimistic scores compared to f_o.

Panel (d) depicts the impact of sampling bias on the scorecard evaluation in four performance metrics. It compares the actual AUC, BS, PAUC and ABR of f_a on the representative holdout sample H (labeled as oracle performance) and the estimated performance of f_a ob-

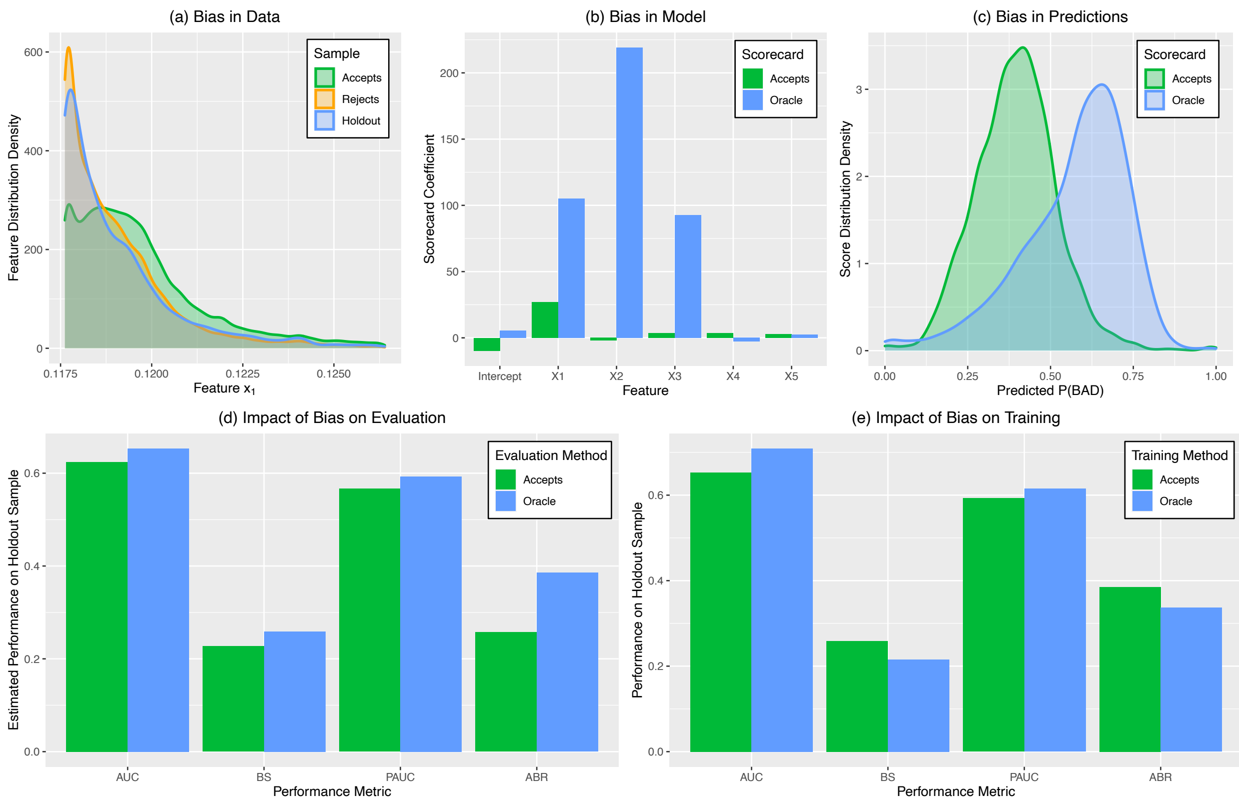

Figure 6.9.3. Sampling Bias Illustration on Real Data

The figure illustrates sampling bias and its adverse effect on the scorecard behavior, training and evaluation on real credit scoring data set. Panel (a) demonstrates the bias in the distribution of one of the features. Panel (b) visualizes the bias in scorecards by comparing their coefficients for the top-five features. Panel (c) shows the bias in scorecard predictions for new loan applications. Panels (d) and (e) depict the impact of bias on scorecard training and evaluation in four performance metrics: the AUC, BS, PAUC and ABR.

tained when evaluating it on a subset of accepts. Performance values are aggregated using leave-one-out validation on the holdout sample. The results suggest that evaluating f_a on accepts provides a misleading estimate in all four performance metrics. This implies that the scorecard evaluation on real data is affected by sampling bias and requires correction. Similarly, panel (e) illustrates the adverse effect of sampling bias on the scorecard training. It compares the AUC, BS, PAUC and ABR of f_a and f_o. In each of the four metrics, the performance of the scorecard deteriorates when training it on a biased set of accepted applications, indicating that the bias also affects model training.

Overall, the results on real data demonstrated in Figure 6.9.3 are in line with the results on synthetic data presented in Figure 6.7.1 in Section 6.7. The sampling bias originates from the fact that the training data of previously accepted applicants is not representative of the borrowers' population. The bias in the data translates to the bias in model parameters and predictions and negatively affects the training and evaluation of scorecards. Bias correction methods could help to improve both of these dimensions.

Experiment I

This section provides the results of the statistical tests performed in Experiment I on real data. To check the statistical significance of the results presented in Table 6.7.1 in Section 6.7, we perform a Friedman's non-parametric rank sum test for performance differences [35]. The null hypothesis of the test is that all evaluation methods have similar performance. The null hypothesis is rejected for all performance measures with p-values below 2.2×10^{-16}. Given that the Friedman test indicates differences in the predictive performance, we proceed with post-hoc tests of pairwise differences between the evaluation methods.

We also use a Nemenyi post-hoc pairwise test, which compares the differences between the average ranks of two methods to the critical difference value determined by the significance level [29]. Figure 6.9.4 depicts the rank differences between the evaluation methods based on the Nemenyi test results. The bold segments connect evaluation techniques for which the rank differences in a given evaluation measure are not statistically significant at a 5% level. The results suggest that the Bayesian evaluation framework outperforms both accepts-based evaluation and importance reweighting.

Experiment II

This section provides the results of the statistical significance tests performed in Experiment II om real data and the ablation study that investigates performance gains from different stages of the BASL framework. Similar to Experiment I, we check the significance of the performance gains presented in Table 6.7.2 in Section 6.7. The null hypothesis of the Friedman test that all bias correction methods have similar performance is rejected for all four performance measures with p-values of each test statistic below 2.2×10^{-16}.

Figure 6.9.5 depicts the rank differences calculated for the pairwise Nemenyi post-hoc

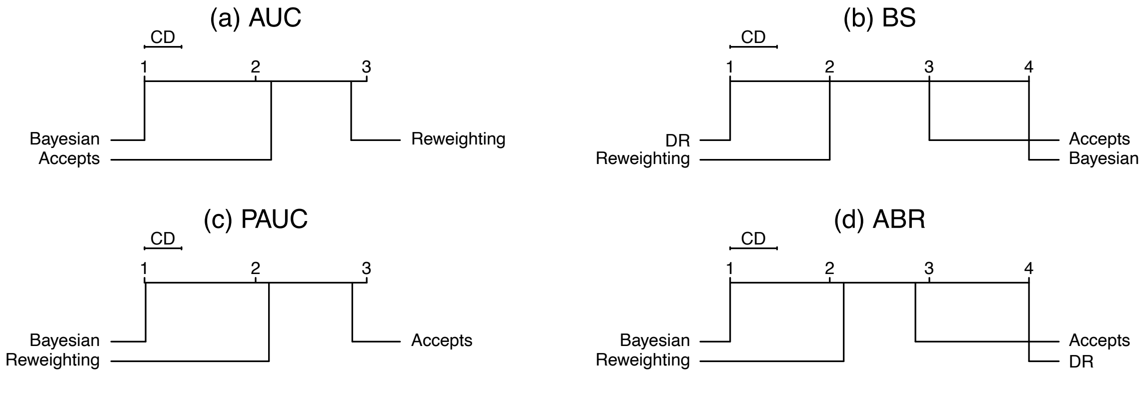

Figure 6.9.4. Experiment I: Critical Difference Plots for Nemenyi Tests

The figure depicts rank differences between evaluation methods. The bold segments connect methods for which the differences are not statistically significant at the 5% level according to the Nemenyi post-hoc pairwise test. Abbreviations: AUC = area under the ROC curve, BS = Brier Score, PAUC = partial AUC on FNR $\in [0, .2]$, ABR = average *bad* rate among accepts at 20-40% acceptance.

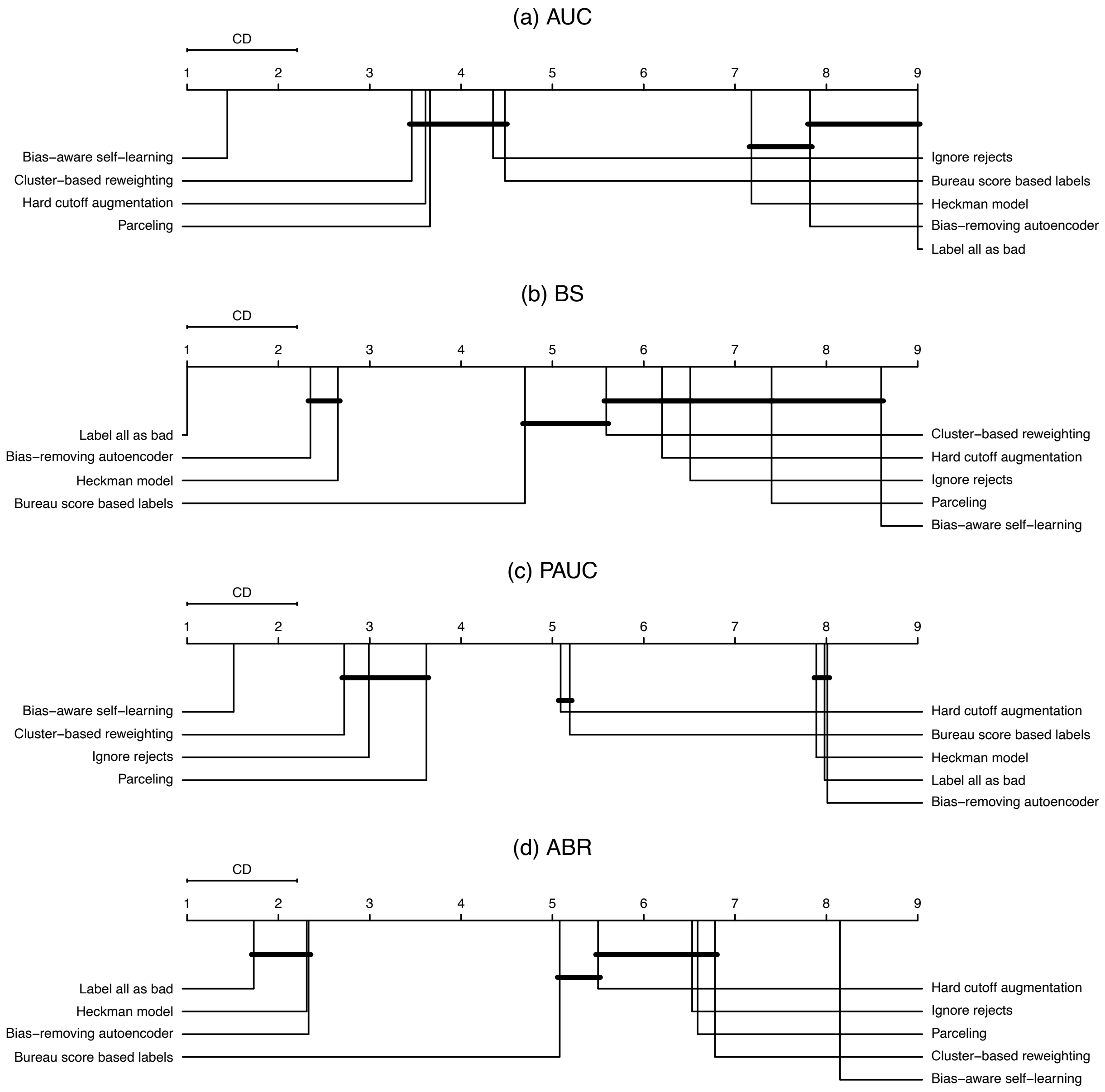

Figure 6.9.5. Experiment II: Critical Difference Plots for Nemenyi Tests

The figure depicts rank differences between evaluation methods. The bold segments connect methods for which the differences are not statistically significant at the 5% level according to the pairwise Nemenyi post-hoc test. Abbreviations: AUC = area under the ROC curve, BS = Brier Score, PAUC = partial AUC on FNR $\in [0, .2]$, ABR = average *bad* rate among accepts at 20-40% acceptance.

tests. As indicated in the figure, BASL outperforms all bias correction benchmarks at a 5% significance level in the AUC, PAUC and ABR. BASL also achieves the best BS, but the BS improvement over the closest competitor, parceling, is not significant at 5% level. In many cases, multiple of the other bias correction benchmarks perform similarly to or worse than ignoring rejects. Parceling and cluster-based reweighting are two methods that tend to come closer to BASL than the other benchmarks in terms of the mean ranks.

Table 6.9.5 provides results of the ablation study of BASL. The table displays incremental

Table 6.9.5. Ablation Study: Gains from Different BASL Steps

Framework extension	AUC	BS	PAUC	ABR	Rank
Traditional self-learning	.8059 (.0010)	.1804 (.0004)	.6868 (.0011)	.2387 (.0020)	4.80
Filter rejects using isolation forest	.8054 (.0011)	.1790 (.0004)	.6981 (.0013)	.2312 (.0022)	3.93
Label rejects with a weak learner	.8134 (.0006)	.1774 (.0002)	.6992 (.0009)	.2294 (.0011)	3.60
Introduce the imbalance multiplier	.8133 (.0006)	.1796 (.0002)	.7026 (.0010)	.2238 (.0012)	3.48
Sampling rejects at each iteration	.8157 (.0006)	.1765 (.0002)	.7035 (.0010)	.2254 (.0013)	2.85
Bayesian metric for early stopping	**.8166** (.0007)	**.1761** (.0003)	**.7075** (.0011)	**.2211** (.0012)	**2.34**

Abbreviations: AUC = area under the ROC curve, BS = Brier Score, PAUC = partial AUC on FNR $\in [0, .2]$, ABR = average *bad* rate among accepts at 20-40% acceptance, rank = average rank across the four evaluation measures.

performance gains from different algorithm steps, starting from traditional self-learning and incorporating the proposed extensions. The extensions make different contributions to the overall performance of BASL.

Overall, incorporating different extensions on top of the traditional self-learning framework improves the the model performance, increasing the PAUC from .6868 to .7075 and the ABR from .2387 to .2211. The largest performance gains in the cost-sensitive metrics are attributed to introducing the filtering stage, which improves the overall rank from 4.80 to 3.93. Gains from implementing the early-stopping mechanism using the Bayesian evaluation framework are observed in all four evaluation metrics, which emphasizes the important role of using a bias-corrected evaluation metric when performing the model selection.

6.9.5 Meta-Parameters of Data Generation and Bias Correction Methods

This appendix provides meta-parameter values of the base classifiers and bias correction methods. We also provide parameters of the data generation process and the acceptance loop used in the simulation study.

Synthetic Data

The data generation process and the acceptance loop have multiple important meta-parameters. Concerning the data generation, we assume the number of mixture components $C = 2$ and set the distribution parameters as follows: $\boldsymbol{\mu}_1^g = (0, 0), \boldsymbol{\mu}_1^b = (2, 1), \boldsymbol{\mu}_2^g = \boldsymbol{\mu}_1^g + \vec{1}$ and $\boldsymbol{\mu}_2^b = \boldsymbol{\mu}_1^b + \vec{1}$. The elements of $\boldsymbol{\Sigma}_c^i$ are drawn from a uniform distribution $\mathcal{U}(0, \sigma_{max})$. We run the acceptance loop for 500 iterations, assuming $n = 100$ and $h = 3{,}000$. In the MAR setup considered in Section 6.7.1, we set $\alpha = .15$, $b = .70$ and $\sigma_{max} = 1$. In the sensitivity analysis presented in Section 6.7.1, we vary α, β and σ_{max} to investigate the boundary conditions affecting the performance of our propositions. In the MNAR setup considered in Section 6.7.1, we assume $\boldsymbol{\mu}_1^g = (0, 0, 0), \boldsymbol{\mu}_1^b = (2, 1, 0.5), \boldsymbol{\mu}_2^g = \boldsymbol{\mu}_1^g + \vec{1}$ and $\boldsymbol{\mu}_2^b = \boldsymbol{\mu}_1^b + \vec{1}$, hiding the feature with the smallest mean difference from the scorecard and using it for overwriting the scorecard

Table 6.9.6. Meta-Parameters of Base Classifiers

Meta-parameter	Candidate values	Selected values (synthetic data)	Selected values (real data)
Maximum number of trees	100, 1,000, 10,000	100	10,000
Early stopping rounds	100, no early stopping	no early stopping	100
Maximum depth	$1, 3, 5$	3	3
Learning rate	$.1, .3$	.1	.1
Bagging ratio	$.8, .9, 1$	.8	.8
Feature ratio	$.8, .9, 1$	.8	.8

predictions. XGB is used as a base classifier for all scoring models. The meta-parameters of XGB on synthetic data are provided in Table 6.9.6.

Concerning the BASL framework, we set the filtering thresholds β to $(.05, 1)$. In the labeling stage, we set $\theta = 2, \rho = .8, \gamma = .01$ and $j_{max} = 3$. We use LR to label the rejected applications and use the Bayesian evaluation framework for early stopping the labeling iterations. To perform the Bayesian evaluation, we set the convergence threshold ϵ to 10^{-6} and specify the number of Monte-Carlo simulations between 10^2 and 10^4. The prior on the labels of rejects denoted as $\mathbf{P}(\mathbf{y}^r|\mathbf{X}^r)$ is obtained by predicting the scores of rejected cases using the accepts-based scoring model and calibrating them using LR.

Real Data

Table 6.9.6 provides the list of the candidate values and the selected values of the meta-parameters of the XGB classifier that is used as a base classifier for all bias correction methods considered in Experiment I and II on the real data. The meta-parameter values are optimized using grid search on a subset of training data.

Table 6.9.7 contains the bias correction methods considered in the empirical comparison, including both training and evaluation strategies. For each bias correction method, we provide a list of their meta-parameters, including a set of candidate values used for the meta-parameter tuning and the values selected after tuning. Two baseline bias correction strategies – ignoring rejects and labeling all rejects as *bad* risks – do not have any meta-parameters and are not included in the table.

6.9.6 Implementation of Benchmarks

This appendix provides further implementation details and additional empirical results for some variants of the bias correction benchmarks not included in the paper. The considered benchmarks include importance reweighting techniques, doubly robust evaluation and the bias-removing autoencoder.

Table 6.9.7. Meta-Parameters of Bias Correction Methods

Bias correction method	Meta-parameter	Candidate values	Selected values
Bias-removing autoencoder	Learning rate	.01	.01
	Number of training epochs	50, 100	100
	Batch size	50, 100	100
	Regularization parameter	$10^{-1}, 10^{-2}, ..., 10^{-5}$	10^{-5}
	Number of hidden layers	3	3
	Bottleneck layer size	$.8k$, k is no. features	$.8k$
Heckman model	Number of features	5, 10, ..., 100	65
	Functional form	probit, logit, linear	linear
Bureau score based labels	Rating of *good* risks	$\{AA\}, \{AA, A\}$	$\{AA, A\}$
	Rating of *bad* risks	$\{D\}, \{C, D\}$	$\{D\}$
Hard cutoff augmentation	Probability threshold	.3, .4, .5	.5
Reweighting	Truncation parameter	.01, .05	.05
	Weight scaling	yes, no	yes
	Density ratio estimation	cluster-based, LSIF, KLIEP	cluster-based
	Number of leaves in DT	100	100
Parceling	Multiplier	1, 2, 3	1
	Number of batches	10	10
Doubly robust evaluation	Truncation parameter	.01, .05	.05
	Weight scaling	yes, no	yes
	Density ratio estimation	cluster-based, LSIF, KLIEP	cluster-based
	Number of leaves in DT	100	100
	Reward prediction model	LR, RF	RF
	Reward prediction threshold	$.1, .2, ..., .9$	.2
Bias-aware self-learning	Filtering thresholds β	(0, 1), (.01, .99), (.01, 1)	(.01, 1)
	Sampling ratio ρ	1, .3	.3
	Labeled percentage γ	.01, .02, .03	.01
	Imbalance parameter θ	1, 2, 3	2
	Max number of iterations j_{max}	5	5
Bayesian evaluation	Max number of trials j_{min}	10^2, 10^3, 10^4	10^2
	Max number of trials j_{max}	10^6	10^6
	Convergence threshold ε	10^{-6}	10^{-6}
	Prior calibration	yes, no	yes

Reweighting

This section focuses on the reweighting techniques considered in this paper. Reweighting tackles sampling bias by estimating importance weights for training examples to rebalance the loss function of the trained algorithm towards examples that are more representative of the population. Given a biased training set D^a and a representative test set $H \subset D$, weights of training examples can be computed as a ratio of two distribution densities: $w(X) = p_H(X)/p_{D^a}(X)$. We focus on the two established families of reweighting techniques: density ratio estimation and cluster-based methods. In addition, we propose and use an alternative weight estimation method that uses isolation forest to produce the importance weights.

We implement two prominent density ratio estimation methods: Kullback-Leibler Importance Estimation Procedure [KLIEP, 93] and Least Square Importance Fitting [LSIF, 50]. These techniques directly estimate the density ratio without explicit estimation of distribution densities $p_H(X)$ and $p_{D^a}(X)$. KLIEP estimates weights by minimizing the Kullback-Leibler divergence between $p_H(X)$ and $w(X)p_{D^a}(X)$. LSIF formulates a least-squares function fitting problem by modeling weights using a linear model: $w(X) = \sum_{l=1}^{b} \alpha_l \phi_l(X)$, where $\alpha = (\alpha_1, \alpha_2, ..., \alpha_b)$ are parameters to be learned from data, and $\{\phi_l(X)\}_{l=1}^{b}$ are basis functions such that $\phi_l(X) \leq 0$ for all $X \in D^a$ and for $l = 1, 2, ..., b$.

The cluster-based method estimates weights based on the empirical frequencies of data examples [25]. The data spits into n clusters, $C = \{C_i\}_{i=1}^{n}$. The example weights within each cluster are computed as a ratio of test and training examples in that cluster: $w(C_i) = |C_i \cup D^a|/|C_i \cup H|$. Following the suggestion of Cortes et al. [25], we use leaves of a fitted decision tree to form the clusters.

Finally, we estimate importance weights using isolation forest, which is a novelty detection algorithm that estimates the normality of each observation by computing the number of splits required to isolate it from the rest of the data [65]. We fit isolation forest $g(X)$ on the attributes of cases $\mathbf{X}^h$ in the representative sample H. Next, we use g to predict similarity scores for the training examples in D^a: $\mathbf{s}^a = g(\mathbf{X}^a \subset D^a)$. The predicted similarity scores can be used to judge the likelihood that a certain example comes from the population distribution. The obtained score vector $\mathbf{s}^a$ is then used as importance weights.

In the domain adaptation literature, importance weights are normally computed based on the comparison between a biased training sample and a representative target test sample. In the credit scoring context, we observe two samples: a labeled set of accepted clients D^a and an unlabeled set of rejected clients D^r. Both accepts and rejects are biased with respect to the general population of loan applicants D. As indicated in Section 6.6, this paper has access to a representative holdout sample consisting of loans that were granted to a random set of applicants without scoring. This sample can be used as a target sample for estimating the importance weights. However, a representative holdout sample is normally not available as it is very costly to obtain. In this case, a representative validation sample can be constructed by combining the applications that were accepted and rejected by a scoring model during the same time interval.

Before computing the importance weights, we drop features that have Spearman's or Pearson's pairwise correlation higher than .95 to reduce dimensionality and lower the potential noise in weight estimates. The resulting data set contains 1,549 features. Next, we perform 4-fold stratified cross-validation on the data of accepted applicants D^a, following the procedure described in Experiment II in Section 6.6. Within the cross-validation framework, we iteratively estimate the importance weights of the training examples among accepts and rejected examples using one of the considered reweighting techniques.

For each reweighting method, we estimate importance weights in multiple distinct ways. For the density ratio estimation methods KLIEP and LSIF, we estimate density ratios on two sets of samples: (i) comparing the data of accepted applicants and a holdout sample; (ii) comparing the data of accepted applicants and a validation sample constructed from both accepts and rejects. The estimated ratios serve as training weights.

For the cluster-based method, we train two decision trees that split the data into clusters: (i) the first variant is trained over the accepted applicants; (ii) the second variant is trained over the holdout sample. Both decision trees are limited to 100 leaves to ensure that we have enough observations in each cluster. We assign each training example to a cluster in accordance with the leave of the decision tree in which this example falls. Next, we use each of the leaves to compute cluster-specific example weights in two ways: (i) as a ratio between the number of accepted examples and holdout examples in the cluster; (ii) as a ratio between the number of accepted examples and validation examples in the cluster. This gives us four cluster-based reweighting methods employing different clustering models and different weight estimation techniques.

Similarly, we use two variants of the isolation forest: (i) trained over the holdout sample; (ii) trained over the validation sample consisting of both accepts and rejects. Next, we use the trained models to produce similarity scores for the accepted examples. The similarity scores are then used as importance weights.

Before using the estimated importance weights for sampling bias correction, we truncate weights to reduce their variance [34]. The weights are truncated to the interval $[\alpha, \frac{1}{\alpha}]$, where α is tuned using grid search. We also normalize the obtained importance weights using min-max scaling.

The resulting weights are used for both scorecard training and evaluation. First, we use the importance weights for scorecard evaluation within Experiment I. The examples in the validation subset that contains accepts and rejects are used to calculate one of the four performance metrics used in the paper: the AUC, BS, PAUC and ABR. The BS and ABR are reweighted by multiplying the corresponding application-level errors by the importance weights. The weighted AUC is calculated using a technique suggested by [52, 44]. Second, within Experiment II, we use the importance weights for the scorecard development. The importance weights of accepts act as training weights when fitting the XGB-based scorecard on the data from the training folds. The meta-parameters of the base classifiers are provided in Table 6.9.6.

Table 6.9.8 provides the extended results of Experiment II in terms of the predictive performance of a corrected scorecard on the holdout sample. The results suggest that only some of the reweighting techniques improve the scorecard performance compared to a model with no weights. Overall, one of the cluster-based methods outperforms the benchmarks from the density ratio estimation techniques and isolation forest based reweighting. The

Table 6.9.8. Performance of Reweighting Techniques

Approach	Weights	Sample	AUC	BS	PAUC	ABR	Rank
No weights	–	–	.7984 (.0010)	.1819 (.0004)	.6919 (.0010)	.2388 (.0019)	5.21
Density ratio	KLIEP	H	.7930 (.0011)	.1874 (.0005)	.6815 (.0014)	.2543 (.0024)	8.52
	KLIEP	V	.7950 (.0011)	.1879 (.0005)	.6775 (.0013)	.2570 (.0023)	8.94
	LSIF	H	.8033 (.0007)	**.1827** (.0004)	.6916 (.0013)	.2372 (.0021)	4.54
	LSIF	V	.8027 (.0008)	.1837 (.0004)	.6936 (.0012)	.2365 (.0019)	4.58
Cluster-based	A/H	H	.7979 (.0010)	.1861 (.0006)	.6895 (.0014)	.2447 (.0024)	6.68
	A/V	H	.7988 (.0010)	.1863 (.0004)	.6848 (.0013)	.2473 (.0021)	7.20
	A/H	V	.8040 (.0008)	.1840 (.0004)	**.6961** (.0012)	**.2346** (.0022)	**4.22**
	A/V	V	.7955 (.0012)	.1844 (.0003)	.6884 (.0013)	.2407 (.0022)	6.35
Isolation forest	SS	V	**.8045** (.0009)	.1837 (.0004)	.6932 (.0013)	.2381 (.0021)	4.43
	SS	V	.8029 (.0009)	.1845 (.0003)	.6925 (.0013)	.2407 (.0021)	5.33

Weights: A/H = no. accepts divided by no. holdout examples, A/V = no. accepts divided by no. of validation examples, SS = similarity score. Sample: sample used to estimate weights, train isolation forest or clustering algorithm; H = holdout, V = validation. Performance measures: AUC = area under the ROC curve, BS = Brier Score, PAUC = partial AUC on FNR $\in [0, .2]$, ABR = average *bad* rate among accepts at 20-40% acceptance rate, rank = the average method rank across the four measures. Standard errors in parentheses.

best performance is achieved when the decision tree that forms the clusters is trained over accepts, whereas the weights are computed as a ratio between the number of accepts and holdout examples in each cluster. Using isolation forest to estimate weights achieves the second-best performance.

The superior performance of the cluster-based reweighting and isolation forest can be explained by the good scalability of tree-based methods in high-dimensional feature spaces. The density ratio estimation methods KLIEP and LSIF produce noisier estimates, which harms the resulting scorecard performance. The cluster-based reweighting demonstrates the best performance when we calculate the importance weights using a time-based validation set constructed of both accepts and rejects. Relying on such a sample is also easier in practice since a representative holdout set is costly to obtain.

Doubly Robust

This section provides additional methodological details on the implementation of the doubly robust off-policy evaluation method [DR, 31]. Due to the differences between the contextual bandit setting considered in the off-policy evaluation literature and the credit scoring setup considered in this paper, using DR for scorecard evaluation requires some adjustments, which we detail below.

In the off-policy evaluation literature, DR is used in a contextual bandit setting. A decision-maker chooses from a set of possible actions and evaluates a policy that determines the assignment of actions. The quality of a policy, or a classifier, is estimated on historical data. In practice, this data is incomplete as every subject has been assigned to exactly one of the possible actions. The reward associated with that action was observed and is available in the data. The (counterfactual) reward corresponding to other actions cannot be observed. To address this, DR combines estimating importance weights, which account for sampling bias in the historical data, with predicting the policy reward for the missing actions. DR produces unbiased estimates if at least one of the two modeled equations is correct [90].

The off-policy evaluation setting resembles the credit scoring setup to some extent. Rewards in the form of repayment outcomes are observed for accepted applications. The credit scorecard acts as a policy that determines the assignment of actions (i.e., acceptance vs. rejection). However, a substantial difference between the off-policy evaluation setup and credit scoring concerns the availability of information on policy rewards. We can measure the policy reward by the classifier loss [31], which indicates the predictive performance of the scorecard. In the off-policy evaluation setup, a reward from one of the possible actions is available for each subject in the historical data. DR is then used to combine the observed rewards for actions with the observed outcome and predicted rewards for the remaining actions, where the outcomes are missing. In credit scoring, rewards are only observed for applications that have been accepted in the past (i.e., assigned to one specific action). No rewards are observed for applications assigned to other actions (i.e., rejected) as the financial institution never learns the repayment behavior of rejects. This implies that we need to predict the missing rewards for all rejects.

A second limitation of DR in a credit scoring context is associated with the measurement of reward as classifier loss. This measurement implies that the use of DR is feasible only if we can calculate the evaluation measure on the level of an individual loan. One exemplary loan-level measure is the BS, which assesses a scorecard by calculating the squared difference between the predicted score and a binary label. However, DR is unable to support non-loan level performance measures, including rank-based indicators. Rank-based indicators such as the AUC are widely used in the credit scoring literature [e.g. 57] and regulatory frameworks such as the Basel Capital Accord highlight their suitability to judge the discriminatory power of scoring systems [e.g. 10, 46]. Lacking support for corresponding performance measures constrains the applicability of DR for credit scoring.

In this paper, we implement DR on both synthetic and real-world data. The labeled data of accepted applications is partitioned into training and validation subsets. The training data is used for training the scorecard that is evaluated with DR. The validation subset provides loan applications used for the evaluation. As with the Bayesian evaluation framework, we append rejects to the validation subset to obtain a representative evaluation sample. The repayment outcomes in the validation set are only available for accepts.

As detailed above, DR includes two main components: calculating propensity scores and predicting missing rewards. The first step involves the estimation of propensity scores or importance weights. For this purpose, we use the same method as for the reweighting benchmarks. The comparison of multiple reweighting procedures is described in detail in

Appendix 6.9.6. In our experiments, cluster-based weights with weight clipping performs best and is used for the DR estimator. We calculate importance weights for both accepted and rejected applications in the validation subset and store them for the next steps of the DR framework.

The second step involves the calculation of policy rewards, which requires producing a vector of classifier losses for each of the applications in the validation set. To calculate rewards, we score applications in the validation subset with the scorecard evaluated by DR. Next, we compute the rewards for accepts. This procedure depends on the considered evaluation metric. For the BS, the reward is simply the squared difference between the risk score predicted by the scorecard and the actual 0-1 application label. The ABR metric only penalizes the type-II error (i.e., accepting a *bad* applicant). Therefore, for the ABR, we compute the policy reward as a binary variable that equals 1 if the application is predicted to be a *good* but is actually a *bad* risk, and 0 otherwise. Calculating the other two performance measures used in the paper – the AUC and PAUC – is not feasible on the application level, so we only use DR with the BS and the ABR.

Apart from rewards for accepted clients, we also require rewards for rejects. However, since the actual labels of rejects are unknown, we have to predict the rewards for such applications. For this purpose, we train a reward prediction model on the accepted cases from the validation subset and use it to predict rewards for rejects. Reward prediction is performed using a random forest (RF) regressor for the BS metric and using an RF classifier for the ABR. Both models process all applicant features to predict rewards. Since the ABR calculation requires binary rewards, we convert classifier scores into the class predictions using a specified threshold, which we tune to minimize the RMSE of the DR performance estimates.

The final step of the DR framework is calculating the estimate of the scorecard performance based on the computed rewards and importance weights. The actual rewards on accepts and predicted rewards on rejects are multiplied with the weights to correct for the sample selection bias. Next, we aggregate the resulting values across all applications in the validation subset. For the BS, this implies averaging the corrected squared differences over the applications. For the ABR, which has acceptance rate as a meta-parameter, we average the corrected binary error indicators over a certain percentage of applications predicted as least risky.

Bias-Removing Autoencoder

In this section, we take a closer look at the performance of the bias-removing autoencoder. The bias-removing autoencoder tackles sampling bias by finding a function that maps features into a new representational space $\mathbf{Z}$ (i.e., $\Phi : \mathbf{X} \rightarrow \mathbf{Z}$) such that distribution of the labeled training data over $\mathbf{Z}$ is less biased and $\Phi(X)$ retains as much information about X as possible. To analyze the performance of this bias correction method in more detail, we compare the predictive performance of four different scoring models that use features coming from the data representations constructed by different autoencoder variants.

The first scoring model $f_a(X)$ serves as a baseline. The model f_a is trained over raw features over a biased sample of previously accepted clients. The next three scorecards are trained over latent features extracted from different autoencoders. The autoencoders are trained using different data samples. First, we train a standard deep stacked autoencoder $a_1(X)$ over $\mathbf{X}^a$ to reconstruct the features of accepted clients. We extract latent features from the bottleneck layer of a_1 and use them for training a new scoring model $f_{a_1}(X)$. The scoring model is, therefore, based on the data representation computed on a biased sample of applicants.

Second, we train the autoencoder $a_2(X)$ with the same architecture as a_1 but using a training sample constructed of both accepted and rejected applicants $\mathbf{X}^a \cup \mathbf{X}^r$. The scoring model $f_{a_2}(X)$ is trained over the latent features extracted from the bottleneck layer of a_2. The extracted features account for the patterns observed on both accepts and rejects, which should improve the performance of the scorecard.

Finally, we train the third autoencoder $a_3(X)$ on $\mathbf{X}^a \cup \mathbf{X}^r$. Compared to a_2, a_3 includes an additional regularization term that accounts for the distribution mismatch similar to Atan et al. [3]. The regularization term penalizes the mismatch between the distributions of latent features on accepted examples and examples in a validation sample consisting of accepts and rejects from the same time window. This helps the autoencoder to derive latent features that are distributed similarly on the two data samples. After training the autoencoder, we train a scoring model $f_{a_3}(X)$ on the extracted feature representation.

In all three cases, we use a stacked deep autoencoder architecture with three hidden layers. The number of neurons is set to $.9k$ on the first and the last hidden layer and $.8k$ on the bottleneck layer, where k is the number of features in the input data. To facilitate convergence, we preprocess the data before training the autoencoder. First, we drop features that have Spearman or Pearson pairwise correlation higher than .95, reducing the number of features to 1,549. Second, we remove outliers by truncating all features at .01 and .99 distribution percentiles. Third, we normalize feature values to lie within $[0, 1]$ interval. Other meta-parameters of the autoencoder are tuned using grid search; the list of candidate values is given in Table 6.9.7. All scoring models use XGB-based classifier as a base model; the meta-parameters are provided in Table 6.9.6.

As a mismatch penalty, we use the Maximum Mean Discrepancy [MMD, 15], which measures a distance between distribution means in a kernel space. The MMD is commonly used as a distribution mismatch measure in the domain adaptation literature [e.g. 80, 68]. The MMD is measured between the latent features of accepts and latent features of the validation sample. The validation sample refers to a time-based representative sample that contains both accepted and rejected clients. The autoencoders only use the features, ignoring the actual labels. Table 6.9.9 reports the results.

Table 6.9.9. Performance of Bias-Removing Autoencoder

Features	Sample	MMD	AUC	BS	PAUC	ABR	Rank
Raw	A	-	**.7984** (.0010)	**.1819** (.0004)	**.6919** (.0010)	**.2388** (.0019)	**1.01**
Latent	A	-	.7006 (.0034)	.2212 (.0008)	.6066 (.0026)	.3385 (.0023)	3.37
Latent	A + R	-	.7177 (.0020)	.2187 (.0004)	.6170 (.0013)	.3328 (.0024)	3.14
Latent	A + R	+	.7304 (.0011)	.2161 (.0004)	.6376 (.0019)	.3061 (.0036)	2.48

Features: features used to train a scorecard (either raw features or latent features extracted from the bottleneck layer of the autoencoder). Sample: training sample of the autoencoder; A = accepts, R = rejects. MMD: whether the MMD penalty is included in the autoencoder loss function. Performance measures: AUC = area under the ROC curve, BS = Brier Score, PAUC = partial AUC on FNR $\in [0, .2]$, ABR = average *bad* rate among accepts at 20-40% acceptance rate, rank = the average strategy rank across the four performance measures. Standard errors n in parentheses.

First, we compare latent features extracted from a_1 trained on accepts and latent features from a_2 trained on both client types. The results suggest that the latter set of features leads to a better predictive performance of the eventual scoring model. Furthermore, including the MMD penalty in the autoencoder loss function allows us to extract features that further improve the scorecard's performance. From this comparison, we can conclude that using data of rejected applications and penalizing the distribution discrepancies helps to find a feature representation that suffers less from sampling bias, which has a positive impact on the performance.

At the same time, comparing the performance of the scoring model f_a trained over the original features of accepts to the scoring model f_{a_1} trained over the latent features of the accepts-based autoencoder, we observe a sharp performance drop in all four evaluation measures. This indicates that the predictive power of the latent features constructed by the autoencoder a_1 is too low compared to that of the original features. The observed information loss is too large to be offset by the performance improvement from using rejects and adding a distribution mismatch regularizer. This can be explained by a high dimensionality of the feature space, which complicates the reconstruction task.

Bibliography

[1] Anderson, B. (2019). Using Bayesian networks to perform reject inference. *Expert Systems with Applications, 137*, 349–356.

[2] Anderson, B., Hardin, J.M. (2013). Modified logistic regression using the EM algorithm for reject inference. *International Journal of Data Analysis Techniques and Strategies, 5*(4), 359–373.

[3] Atan, O., Jordon, J., van der Schaar, M. (2018). Deep-treat: Learning optimal per-

sonalized treatments from observational data using neural networks. *Proc. 32nd AAAI Conference on Artificial Intelligence.*

[4] Athey, S., Wager, S. (2021). Policy learning with observational data. *Econometrica, 89*(1), 133–161.

[5] Baesens, B., Setiono, R., Mues, C., Vanthienen, J. (2003). Using neural network rule extraction and decision tables for credit-risk evaluation. *Management Science, 49*(3), 312–329.

[6] Ban, G.Y., Rudin, C. (2019). The big data newsvendor: Practical insights from machine learning. *Operations Research, 67*(1), 90–108.

[7] Banasik, J., Crook, J. (2005). Credit scoring, augmentation and lean models. *Journal of the Operational Research Society, 56*(9), 1072–1081.

[8] Banasik, J., Crook, J. (2007). Reject inference, augmentation, and sample selection. *European Journal of Operational Research, 183*(3), 1582–1594.

[9] Banasik, J., Crook, J., Thomas, L. (2003). Sample selection bias in credit scoring models. *Journal of the Operational Research Society, 54*(8), 822–832.

[10] Basel Committee on Banking Supervision (2005). Studies on the validation of internal rating systems. *BIS Working Paper Series* 14.

[11] Bhat, G., Ryan, S.G., Vyas, D. (2019). The implications of credit risk modeling for banks' loan loss provisions and loan-origination procyclicality. *Management Science, 65*(5), 2116–2141.

[12] Biatat, V.A.D., Crook, J., Calabrese, R., Hamid, M. (2021). Enhancing credit scoring with alternative data. *Expert Systems with Applications, 163*, 113766.

[13] Bickel, S., Brückner, M., Scheffer, T. (2009). Discriminative learning under covariate shift. *Journal of Machine Learning Research, 10*(9).

[14] Blitzer, J., McDonald, R., Pereira, F. (2006). Domain adaptation with structural correspondence learning. *Proc. 2006 Conference on Empirical Methods in Natural Language Processing*, 120–128.

[15] Borgwardt, K.M., Gretton, A., Rasch, M.J., Kriegel, H.P., Schölkopf, B., Smola, A.J. (2006) Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics, 22*(14), e49–e57.

[16] Boyes, W.J., Hoffman, D.L., Low, S.A. (1989). An econometric analysis of the bank credit scoring problem. *Journal of Econometrics, 40*(1), 3–14.

[17] Briceño, J., Cruz-Ramírez, M., Prieto, M., Navasa, M., De Urbina, J.O., Orti, R., Gómez-Bravo, M.Á., Otero, A., Varo, E., Tomé, S., et al. (2014). Use of artificial intelligence as an innovative donor-recipient matching model for liver transplantation: results from a multicenter spanish study. *Journal of Hepatology, 61*(5), 1020–1028.

[18] Bruzzone, L., Marconcini, M. (2010). Domain adaptation problems: A DASVM classification technique and a circular validation strategy. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 32*(5), 770–787.

[19] Bücker, M., van Kampen, M., Krämer, W. (2012). Reject inference in consumer credit scoring with nonignorable missing data. *Journal of Banking & Finance, 37*(3), 1040–1045.

[20] Caseiro, R., Henriques, J.F., Martins, P., Batista, J. (2015). Beyond the shortest path: Unsupervised domain adaptation by sampling subspaces along the spline flow. *Proc. 28th IEEE Conference on Computer Vision and Pattern Recognition*, 3846–3854.

[21] Chen, G.G., Astebro, T. (2001). The economic value of reject inference in credit scoring. *Proc. 7th Credit Scoring and Credit Control Conference*, 309–321.

[22] Chen, G.G., Åstebro, T. (2012). Bound and Collapse Bayesian reject inference for credit scoring. *Journal of the Operational Research Society* 63(10), 1374–1387.

[23] Chen, M., Weinberger, K.Q., Blitzer, J. (2011). Co-training for domain adaptation. *Advances in Neural Information Processing Systems, 24*, 2456–2464.

[24] Chen, X., Monfort, M., Liu, A., Ziebart, B.D. (2016). Robust covariate shift regression. *Artificial Intelligence and Statistics*, 1270–1279.

[25] Cortes, C., Mohri, M., Riley, M., Rostamizadeh, A. (2008). Sample selection bias correction theory. *Proc. 19th International Conference on Algorithmic Learning Theory*, 38–53.

[26] Crook, J., Banasik, J. (2004). Does reject inference really improve the performance of application scoring models? *Journal of Banking & Finance* 28(4), 857–874.

[27] Daumé III, H (2009). Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815.*

[28] Daumé III, H., Marcu, D. (2006). Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research, 26*, 101–126.

[29] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30.

[30] Duan, L., Xu, D., Tsang, I.W.H. (2012). Domain adaptation from multiple sources: A domain-dependent regularization approach. *IEEE Transactions on Neural Networks and Learning Systems, 23*(3), 504–518.

[31] Dudík, M., Erhan, D., Langford, J., Li, L. (2014). Doubly robust policy evaluation and optimization. *Statistical Science, 29*(4), 485–511.

[32] Feelders, A.J. (2000). Credit scoring and reject inference with mixture models. *Intelligent Systems in Accounting, Finance and Management Decision* 9(1), 1–8.

[33] Fogarty, D.J. (2006). Multiple imputation as a missing data approach to reject inference on consumer credit scoring. *Interstat, 41*, 1–41.

[34] Freedman, D.A., Berk, R.A. (2008). Weighting regressions by propensity scores. *Evaluation Review, 32*(4), 392–409.

[35] Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association, 32*(200), 675–701.

[36] Gong, B., Grauman, K., Sha, F. (2013). Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. *Proc. 30th International Conference on Machine Learning*, 222–230.

[37] Gong, B., Shi, Y., Sha, F., Grauman, K. (2012). Geodesic flow kernel for unsupervised domain adaptation. *Proc. 25th IEEE Conference on Computer Vision and Pattern Recognition*, 2066–2073.

[38] Gopalan, R., Li, R., Chellappa, R. (2011). Domain adaptation for object recognition: An unsupervised approach. *Proc. 13th International Conference on Computer Vision*, 999–1006.

[39] Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B., Smola, A. (2012). A kernel two-sample test. *The Journal of Machine Learning Research* 13(1), 723–773.

[40] Gu, S., Kelly, B., Xiu, D. (2020). Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies, 33*(5), 2223–2273.

[41] Gunnarsson, B.R., Vanden Broucke, S., Baesens, B., Óskarsdóttir M., Lemahieu W. (2021). Deep learning for credit scoring: Do or don't? *European Journal of Operational Research, 295*(1), 292–305.

[42] Heckman, J.J. (1979). Sample selection bias as a specification error. *Econometrica, 47*(1), 153–161.

[43] Hilscher, J., Wilson, M. (2016). Credit ratings and credit risk: Is one measure enough? *Management Science, 63*(10), 3414–3437.

[44] Hocking, T.D. (2020). *WeightedROC: Fast, Weighted ROC Curves. R package version 2020.1.31.* URL `https://CRAN.R-project.org/package=WeightedROC`. Accessed 2021-12-01.

[45] Huang J., Gretton, A., Borgwardt, K., Schölkopf, B., Smola, A. (2006). Correcting sample selection bias by unlabeled data. *Advances in Neural Information Processing Systems, 19*, 601–608.

[46] Irwin, R.J., Irwin, T.C. (2012). Appraising credit ratings: Does the CAP fit better than the ROC? *IMF Working Paper* 12/122.

[47] Jagtiani, J., Lemieux, C. (2019). The roles of alternative data and machine learning in fintech lending: Evidence from the LendingClub consumer platform. *Financial Management, 48*(4), 1009–1029.

[48] Joachims, T., Swaminathan, A., Schnabel T. (2017). Unbiased learning-to-rank with biased feedback. *Proc. 10th ACM International Conference on Web Search and Data Mining*, 781–789.

[49] Joanes, D.N. (1993). Reject inference applied to logistic regression for credit scoring. *IMA Journal of Management Mathematics, 5*(1), 35–43.

[50] Kanamori, T., Hido, S., Sugiyama, M. (2009). A least-squares approach to direct importance estimation. *Journal of Machine Learning Research* 10(Jul), 1391–1445.

[51] Kang, Y., Jia, N., Cui, R., Deng, J. (2021). A graph-based semi-supervised reject inference framework considering imbalanced data distribution for consumer credit scoring. *Applied Soft Computing, 105*, 107259.

[52] Keilwagen, J., Grosse, I., Grau, J. (2014). Area under precision-recall curves for weighted and unweighted data. *PloS one, 9*(3), e92209.

[53] Kim, A., Cho, S.B. (2019). An ensemble semi-supervised learning method for predicting defaults in social lending. *Engineering Applications of Artificial Intelligence, 81*, 193–199.

[54] Kim, Y., Sohn, S.Y. (2007). Technology scoring model considering rejected applicants and effect of reject inference. *Journal of the Operational Research Society, 58*(10), 1341–1347.

[55] Kozodoi, N., Katsas, P., Lessmann, S., Moreira-Matias, L., Papakonstantinou, K. (2019). Shallow self-learning for reject inference in credit scoring. *Proc. European Conference on Machine learning and Knowledge Discovery in Databases*, 516–532.

[56] Kügelgen, J., Mey, A., Loog, M. (2019). Semi-generative modelling: Covariate-shift adaptation with cause and effect features. *Proc. 22nd International Conference on Artificial Intelligence and Statistics*, 1361–1369.

[57] Lessmann, S., Baesens, B., Seow, H.V., Thomas, L.C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research, 247*(1), 124–136.

[58] Levatić, J., Ceci, M., Kocev, D., Džeroski, S. (2017). Self-training for multi-target regression with tree ensembles. *Knowledge-Based Systems* 123, 41–60.

[59] Li, Z., Tian, Y., Li, K., Zhou, F., Yang, W. (2017). Reject inference in credit scoring using semi-supervised support vector machines. *Expert Systems with Applications, 74*, 105–114.

[60] Lin, Y., Lee, Y., Wahba, G. (2002). Support vector machines for classification in nonstandard situations. *Machine Learning, 46*(1-3), 191–202.

[61] Little, R.J. (1988). A test of missing completely at random for multivariate data with missing values. *Journal of the American Statistical Association* 83(404), 1198–1202.

[62] Little, R.J., Rubin, D.B. (2019). *Statistical analysis with missing data.* John Wiley & Sons.

[63] Liu, A., Fathony, R., Ziebart, B.D. (2017). Kernel robust bias-aware prediction under covariate shift. *arXiv preprint arXiv:1712.10050.*

[64] Liu, A., Ziebart, B. (2014). Robust classification under sample selection bias. *Advances in neural information processing systems, 27*, 37–45.

[65] Liu, F.T., Ting, K.M., Zhou, Z.H. (2008). Isolation Forest. *Proc. 8th IEEE International Conference on Data Mining*, 413–422.

[66] Liu, Y., Li, X., Zhang, Z. (2020). A new approach in reject inference of using ensemble learning based on global semi-supervised framework. *Future Generation Computer Systems, 109*, 382–391.

[67] Long, M., Wang, J., Ding, G., Pan, S.J., Yu, P.S. (2014). Adaptation regularization: A general framework for transfer learning. *IEEE Transactions on Knowledge and Data Engineering, 26*(5), 1076–1089.

[68] Long, M., Wang, J., Ding, G., Sun, J., Yu, P.S. (2014). Transfer joint matching for unsupervised domain adaptation. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1410–1417.

[69] Loog, M. (2012). Nearest neighbor-based importance weighting. *Proc. 22nd IEEE International Workshop on Machine Learning for Signal Processing*, 1–6.

[70] Maldonado, S., Paredes, G. (2010). A semi-supervised approach for reject inference in credit scoring using SVMs. *Proc. 10th Industrial Conference on Data Mining*, 558–571.

[71] Malistov, A., Trushin, A. (2019). Gradient boosted trees with extrapolation. *Proc. 18th IEEE International Conference on Machine Learning and Applications*, 783–789.

[72] Mancisidor, R.A., Kampffmeyer, M., Aas, K., Jenssen, R. (2020). Deep generative models for reject inference in credit scoring. *Knowledge-Based Systems* 105758.

[73] Marlin, B.M., Zemel, R.S. (2009). Collaborative prediction and ranking with non-random missing data. *Proc. 3rd ACM Conference on Recommender Systems*, 5–12.

[74] Marra, G., Radice, R., Filippou, P. (2017). Regression spline bivariate probit models: a practical approach to testing for exogeneity. *Communications in Statistics-Simulation and Computation, 46*(3), 2283–2298.

[75] Marshall, A., Tang, L., Milne, A. (2010). Variable reduction, sample selection bias and bank retail credit scoring. *Journal of Empirical Finance* 17(3), 501–512.

[76] Martens, D., Baesens, B., van Gestel, T., Vanthienen, J. (2007). Comprehensible credit scoring models using rule extraction from support vector machines. *European Journal of Operational Research, 183*(3), 1466–1476.

[77] Meng, C.L., Schmidt, P. (1985). On the cost of partial observability in the bivariate probit model. *International Economic Review*, 71–85.

[78] Nguyen, H.T. (2016). Reject inference in application scorecards: evidence from France, Working paper, Paris Nanterre University, Paris.

[79] Niculescu-Mizil, A., Caruana, R. (2005). Obtaining calibrated probabilities from boosting. *Proc. 21st Conference on Uncertainty in Artificial Intelligence*, 28–33.

[80] Pan, S.J., Tsang, I.W., Kwok, J.T., Yang, Q. (2011). Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks* 22(2), 199–210.

[81] Rosenbaum, P.R., Rubin, D.B. (1983). The central role of the propensity score in observational studies for causal effects. *Biometrika, 70*(1), 41–55.

[82] Sadhwani, A., Giesecke, K., Sirignano, J. (2020). Deep learning for mortgage risk. *Journal of Financial Econometrics, 19*(2), 313–368.

[83] Saenko, K., Kulis, B., Fritz, M., Darrell, T. (2010). Adapting visual category models to new domains. *Proc. 11th European Conference on Computer Vision*, 213–226 (Springer).

[84] Satpal, S., Sarawagi, S. (2007). Domain adaptation of conditional probability models via feature subsetting. *Proc. 11th European Conference on Principles of Data Mining and Knowledge Discovery*, 224–235.

[85] Shen, F., Zhao, X., Kou, G. (2020). Three-stage reject inference learning framework for credit scoring using unsupervised transfer learning and three-way decision theory. *Decision Support Systems, 137*, 113366.

[86] Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference, 90*(2), 227–244.

[87] Simester, D., Timoshenko, A., Zoumpoulis, S.I. (2020). Efficiently evaluating targeting policies: Improving on champion vs. challenger experiments. *Management Science, 66*(8), 3412–3424.

[88] Simester, D., Timoshenko, A., Zoumpoulis, S.I. (2020). Targeting prospective customers: Robustness of machine-learning methods to typical data challenges. *Management Science, 66*(6), 2495–2522.

[89] Sirignano, J., Giesecke, K. (2019). Risk analysis for large pools of loans. *Management Science, 65*(1), 107–121.

[90] Su, Y., Dimakopoulou, M., Krishnamurthy, A., Dudík M. (2020). Doubly robust off-policy evaluation with shrinkage. *Proc. 37th International Conference on Machine Learning*, 9167–9176.

[91] Sugiyama, M., Krauledat, M., Müller, K.R. (2007). Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research, 8*, 985–1005.

[92] Sugiyama, M., Müller, K.R. (2006). Input-dependent estimation of generalization error under covariate shift. *Statistics & Decisions, 23*(4), 249–279.

[93] Sugiyama, M., Nakajima, S., Kashima, H., Von Buenau, P., Kawanabe, M. (2007). Direct importance estimation with model selection and its application to covariate shift adaptation. *Advances in Neural Information Processing Systems, 7*, 1433–1440.

[94] Sugiyama, M., Ogawa, H. (2001). Subspace information criterion for model selection. *Neural Computation, 13*(8), 1863–1889.

[95] Sun, B., Feng, J., Saenko, K. (2016). Return of frustratingly easy domain adaptation. *Proc 30th AAAI Conference on Artificial Intelligence.*

[96] Tian, Y., Yong, Z., Luo, J. (2018). A new approach for reject inference in credit scoring using kernel-free fuzzy quadratic surface support vector machines. *Applied Soft Computing, 73*, 96–105.

[97] Van Vlasselaer, V., Eliassi-Rad, T., Akoglu, L., Snoeck, M., Baesens, B. (2017). Gotcha! Network-based fraud detection for social security fraud. *Management Science, 63*(9), 3090–3110.

[98] Verstraeten, G., Van den Poel, D. (2005). The impact of sample bias on consumer credit scoring performance and profitability. *Journal of the Operational Research Society, 56*, 981–992.

[99] Walter, S.D. (2005). The partial area under the summary ROC curve. *Statistics in Medicine, 24*(13), 2025–2040.

[100] Wang, F., Rudin, C. (2017). Extreme dimension reduction for handling covariate shift. *arXiv preprint arXiv:1711.10938.*

[101] Wei, Y., Yildirim P., Van den Bulte, C., Dellarocas, C. (2016). Credit scoring with social network data. *Marketing Science, 35*(2), 234–258.

[102] Wu, I.D., Hand, D.J. (2007). Handling selection bias when choosing actions in retail credit applications. *European Journal of Operational Research* 183(3), 1560–1568.

[103] Xia, Y. (2019). A novel reject inference model using outlier detection and gradient boosting technique in peer-to-peer lending. *IEEE Access* 7, 92893–92907.

[104] Xia, Y., Yang, X., Zhang, Y. (2018). A rejection inference technique based on contrastive pessimistic likelihood estimation for P2P lending. *Electronic Commerce Research and Applications, 30*, 111–124.

[105] Yang, J., Yan, R., Hauptmann, A.G. (2007). Adapting SVM classifiers to data with shifted distributions. *Proc. 7th IEEE International Conference on Data Mining Workshops (ICDMW 2007)*, 69–76.

[106] Zadrozny, B. (2004). Learning and evaluating classifiers under sample selection bias. *Proc. 21st International Conference on Machine learning*, 903–910.