# Combating Sampling Bias: A Self-Training Method in Credit Risk Models

**Jingxian Liao**[1,3*], **Wei Wang**[1], **Jason Xue**[1], **Anthony Lei**[2], **Xue Han**[1], **Kun Lu**[1]

[1]Intuit AI+Data, Intuit, Inc.
[2]QuickBooks Capital, Intuit, Inc.
[3]Department of Computer Science, University of California Davis
jxliao@ucdavis.edu, {wei_wang3, jason_xue, anthony_lei, lulu_han, kun_lu}@intuit.com

## Abstract

A significant challenge in credit risk models for underwriting is the presence of bias in model training data. When most credit risk models are built using only applicants who had been funded for credit, such non-random sampling predominantly influenced by credit policymakers and previous loan performances may introduce sampling bias to the models, and thus alter their prediction of default on loan repayment when screening applications from prospective borrowers. In this paper, we propose a novel data augmentation method that aims to identify and pseudo-label parts of the historically declined loan applications to mitigate sampling bias in the training data. We also introduce a new measure to assess the performance from the business perspective, loan application approval rates at various loan default rate levels. Our proposed methods were compared to the original supervised learning model and the traditional sampling issue remedy techniques in the industry. The experiment and early production results from deployed model show that self-training method with calibrated probability as data augmentation selection criteria improved the ability of credit scoring to differentiate default loan applications and, more importantly, can increase loan approval rate up to 8.8%, while keeping similar default rate comparing to baselines. The results demonstrate practical implications on how future underwriting model development processes should follow.

## Introduction

Credit risk models are tools that financial institutions design to guide lending decisions for businesses or individuals. Such model predicts the probability of default, i.e., applicants' probability of not repaying their debts, from collected financial information during the application stage. It is a binary classification model that separates bad borrowers from good ones. Traditional credit risk models are trained with only a part of loan applicants that financial institutions approved since repayment performance is only available for funded loans. Accepted applicants are already screened by the credit risk models and manual checks during the underwriting process. In comparison, the entire application population includes rejected applicants whose actual repayment behavior is unknown and potential applicants who never apply. Therefore, from data sampling perspective, the training samples from accepted applicants are biased in that the through-the-door population at the time of credit underwriting could be significantly different from the larger unknown population (Eisenbeis 1977; Feelders 2000). Although it is difficult to consider potential applicants since no financial information is provided, this paper proposes approaches to address the sampling bias issue by inferring rejected applicants and augmenting representativeness of the training samples. This technique in the lending domain is known as reject inference (Siddiqi 2003; Montrichard 2007).

The reject inference refers to techniques that resolve sampling bias through inferring labels for rejects (Kozodoi et al. 2019), which share similar ideas with semi-supervised learning. They combine accepted applicants with their repayment and rejected applicants with estimated performance into inferred datasets and generate reject inference scoring models. Recent ML based works have proposed new models to assign labels to the rejects from the angle of semi-supervised learning, such as semi-supervised SVMs, self-learning, and K-prototype clustering (Li et al. 2017, 2020; Kozodoi et al. 2019). One common practice used in the industry is to obtain external loan performance data from credit bureaus for rejected applicants, though it is relatively costly. Another well-known strategy, fuzzy augmentation, assigns labels to the rejects based on the scoring model trained by accepted applicants with adjustment made on sample weights (Montrichard 2007). However, empirical results and reviews agreed that the common reject inference methods now do not outperform credit risk models with only accepted loan performances (Hand and Henley 1993; Ehrhardt et al. 2021).Moreover, recent model-based methods like clustering have application limitations, such as having good performances on low dimensional data only according to theoretical findings. Datasets used in these studies are usually oversimplified as low dimension and relatively small in size as well (Li et al. 2017, 2020).

The contribution of this paper is two-fold. **First**, we propose a novel technique to label the rejected applicants and augment the training set for sampling representativeness. We applied the self-training method for rejected application labeling, with variations on the choice of confident unlabeled predictions added to the training set. Specifically, we intro-

*This work was done when Jingxian Liao interned at Intuit AI+Data

duced probability calibration and Trust Score as confidence models to select the most confident predictions (Triguero, García, and Herrera 2015; Jiang et al. 2018; Niculescu-Mizil and Caruana 2005). **Second**, we introduce a new business-related measure (denoted as *approval rate*) to evaluate the performance of reject inference methods. By controlling the default rate, estimated approval rate measures the percentage of applicants approved as an estimated business Key Performance Indicator (KPI). This measure considers both accepted label accuracy and also the application population. It provides us a unique metric for domain-specific evaluation.

## Background

### Application Outcomes and Loan Outcomes

The result of a loan application is approval or reject after lending institutions evaluate the credit risks of repayment, with rare exceptions. And the outcome of a credit decision for an approved loan is not fully known until the loan has matured and either the full amount due is repaid in the expected time or what is repaid is a partial amount and/or over a much longer period of time. We define a loan to be in *good standing* (labeled as Good) when timely payments are being made or payments are less than 60 days past due. The default size is much smaller than the loans with good standing with the help of credit scoring in practice, which leads to the imbalance of data. Using this definition for our discussion, we will simplify application results and loan outcomes as follows:

- *Loan application accepted* – Loan applications that are approved and taken by the borrowers.
- *Loan application rejected* – Loan applications that lenders declined because of lack of creditworthiness deemed by the lenders.
- *Loan with Good Outcome* – Loans are all those loans still in good standing which will mature in 30 days plus all those loans already repaid in full.
- *Loan with Bad Outcome* – Loans are all the rest: the ones that are delinquent (60+ days past due) or not fully repaid (write-offs due to charge off).

## Methods

This section presents our reject inference method: Self-training method, which combines a self-training algorithm and a pseudo-label confidence model.

Consider a set of $n$ loan applications $x_1, x_2, \cdots, x_n \in R^k$ where k is the number of features. This set includes $m$ accepted applications $x_1, x_2, \cdots, x_m \in X_a$ with corresponding labels $y_1, y_2, \cdots, y_m \in \{\text{Good, Bad}\}$ and consists of $x_{m+1}, \cdots, x_n \in X_u$ whose labels are unknown. The credit scoring model trained with $X_a$ only is denoted as Known Good/Bad (KGB) model. To mitigate sampling bias, our method assigns labels to partial unlabeled applications $X_u$, and combines accepted data and pseudo-labeled data into inferred data sets to represent the whole application population and update credit scoring models. The scoring model with inferred data as training set is denoted as reject inference model (RI model).

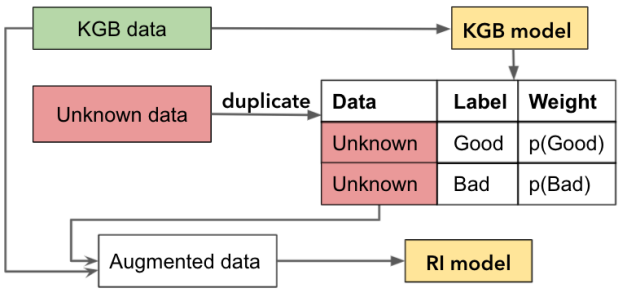


Figure 1: Flowchart of Fuzzy Augmentation.

### Fuzzy Augmentation

Fuzzy augmentation is a representative of popular reject inference techniques discussed, and we will use it as one benchmark (Hsia 1978). It involves assigning labels to unlabeled data based on the KGB model and retrain to get fuzzy augmentation model (FA) (Montrichard 2007), as shown in Figure 1. It assigns unknown data as being partial Good and partial Bad by labels and weights. Every application in $X_u$ is duplicated as two records with two labels $y$: (1) $y_1 = \text{Good}$ with weight $p(\text{Good})$; and (2) $y_2 = \text{Bad}$ with weight $p(\text{Bad})$. The weights $p(\text{Good})$ and $p(\text{Bad})$ are predicted probabilities based on KGB model. The sum of two weights is equal to 1. And accepted applications are also weighted by 1. Then the FA model is constructed on weighted data.

### Self-training with Confidence Model

Figure 2 is an overview of our proposed self-training method pipeline. It starts with training an initial model (also the KGB model in the first iteration) on the accepted data $X_a$ and uses it to predict all the unlabeled data. Then, a confidence model is introduced to filter the most confident predictions whose labels are either good or bad in unlabeled data $X_u$ with fine-tuned thresholds. The selected unlabeled data are labeled by the predictions, and the training set is augmented with new labeled data, denoted as $X_a^1$. Then RI model is retrained with labeled data $X_a^1$. This process is repeated, and self-training model and confidence model update every iteration along with labeled data $X_a^j$. It will stop until no confident applications are identified from the confidence model or reaching a pre-defined number of rounds as stopping criteria.

We applied two confidence models to accommodate the attributes of different algorithms for reject inference: Trust Score (Jiang et al. 2018) and probability calibration (Niculescu-Mizil and Caruana 2005). The traditional self-training selects confident predictions whose prediction probabilities $p$ satisfy $p > \alpha$ or $p < 1 - \alpha$, where $a$ is a probability threshold. However, a part of machine learning classifiers, such as Naive Bayes, SVM, and Random Forest, tend to yield a characteristic sigmoid-shaped distortion in predicted probabilities. Our probability calibration confidence model adds isotonic probability calibration

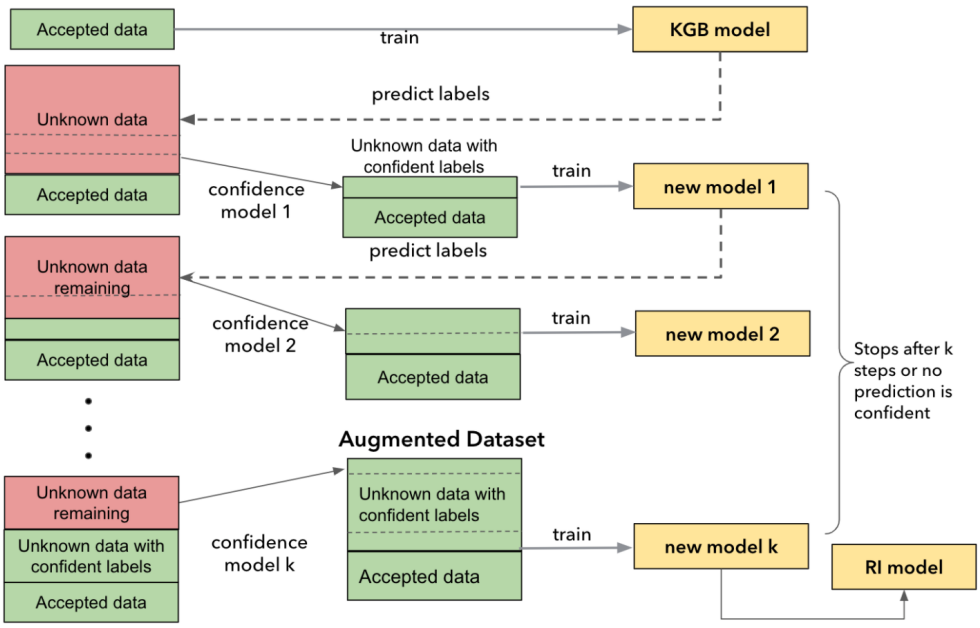


Figure 2: Flowchart of self-training method with confidence model for reject inference.

(Niculescu-Mizil and Caruana 2005) and uses calibrated probabilities to filter confident predictions. The calibrated probability is estimated by a non-parametric isotonic regressor which outputs a step-wise non-decreasing function. It is well-known that it can more accurately represent the confidence level of the probability output. From empirical results, probability calibration shows a significant improvement in maximum margin methods, such as XGBoost. Trust Score model, on the other hand, provides prediction accuracy from the nearest neighboring approach (Jiang et al. 2018). It preselects a high-density data range for each class. Then a trust score is defined as follows to evaluate the prediction: for a predicted test label, the trust score is the ratio between the distance from the test label to the nearest class different from the predicted label class and the distance to the predicted label class within the data range. In this work, the score is based on 5% of the instances from each class. A high score implies high prediction accuracy since the predicted case is close to labeled data with the same label class. It is an alternative to algorithms' own confidence scores from the initial feature space and validation set.

## Experiments and Results

### Data

Our experiment was carried out using loan data from Intuit lending business which offers business loans to its small business accounting software (QuickBooks Online) users since 2017. These loans are repaid weekly, bi-weekly or monthly over a period of six, nine, twelve or eighteen months. Since its inception, hundreds of thousands of loan applications have been submitted, and tens of thousands of loans have been issued. Over a quarter of issued loans have reached maturity, meaning they were either being paid back or defaulted. Those issued loans still in the process of repayment are not included in the credit risk models due to a lack of loan performance history.

For the experiment, Intuit provided around 50,000 random and anonymous samples of loan applications to ensure the representativeness of the population.

Numbers of features are derived corresponding to account balance patterns, cash flow trends, composition of recurring liabilities, seasonality and other spending patterns, frequency of negative financial events such as overdrafts and late payments, et cetera. We will not discuss here hundreds of features that are extracted from bank transactions and how credit bureau data was fetched and processed through third-party API and internal data pipeline, apart for noting that this kind of data is intrinsically noisy. Some of the noises are introduced by information representation and transmission of bank data, inaccurate recording of business bureau data, and significant variability due to the differences in the nature of business among loan applicants.

After feature engineering, the entire dataset was split into a training set and a test set according to the loan application date. For a fair evaluation of a more representative application population, we augmented the labeled test data by assigning labels to part of rejected applicants according to their external credit history from credit bureaus as ground-truth. Therefore, the labeled subset is a combination of internal loans with their performances and rejected applications with estimated labels from their bureau credit history. About 13% of the labeled test set are rejected applications with estimated labels from their bureau credit history. We set stringent labeling criteria to eliminate false positive matches, such as considering external credit accounts history in a narrow time window around application time to guarantee temporal accuracy, looking up loans whose types and days past

| Bureau \ Intuit | Good | Bad |
|---|---|---|
| Good | 91% | $\sim 0\%$ |
| Bad | 3% | 6% |

Table 1: Label matching between Bureau and Intuit. Bureau data are shown in rows and Intuit data are shown in columns. Percentages are shown as % of the total number of loans with labels.

due are similar to our loan population exclusively.

To further validate the quality of ground-truth labeling on the test set, we calculated the confusion matrix between existing internal labeled loan labels and their corresponding labels assigned from credit bureau data. Results show that the matching quality is satisfactory, as shown in Table 1 where 97% of the labels were matched.

## Credit Risk Models

We applied two credit risk models in this experiment, weighted logistic regression and gradient boosted tree algorithm (XGBoost). Logistic regression is one of the common methods used in credit scoring since it is easy to implement and interpret, and it has been widely used to compare reject inference methods with authentic and simulated data (Nguyen et al. 2016). Note that the size of Bad class is much smaller than that of Good. In order to balance the data distribution, we applied weighted logistic regression instead of the default. The weight is the inverse of the label size ratio.

On the other hand, previous work (Wang et al. 2020) found that gradient boosted tree algorithm (XGBoost) provided the best model performance among several algorithms for credit risk scoring, and simultaneously monotonic constraints (DMLC/xgboost 2016) on inputs can provide explanations on the predicted score in conjunction with Shapley values (Lundberg and Lee 2017). Best hyperparameters used in our XGBoost is determined by Amazon Sagemaker XGBoost hyperparameter tuning using Bayesian search (Amazon-Sagemaker 2020).

## Evaluation Metrics

The benchmarks and our new methods are tested on the same test set to ensure a fair comparison.

**AUC-ROC (AUC) and K-S** are used to compare the performances in this experiment because of data imbalance. Besides the regular AUC metric, K-S statistic is a metric between 0 and 1 that measures the maximum separation between the cumulative distribution of the two classes (Bradley 1997; Smirnov 1948; Kolmogorov 1933). Note that both metrics do not depend on the selection of classification thresholds, making them attractive as evaluation metrics in credit risk domain.

**Approval Rate** To supplement domain-independent evaluation metrics, we introduce an innovative evaluation metric from a business KPI perspective, **approval rate**. In general, when more applications are approved, more loans with bad outcome will be introduced. For maximum profit and risk control, lending institutions prefer to extend their customer population while keeping controllable potential loan defaults. Therefore, approval rate is designed to measure this trade-off in credit scoring model performance ad hoc analysis. It is analogous to hypothesis testing – it controls type II error rate and reports the fraction under the null hypothesis (in our case, loans with good label). From the lending business perspective, it predicts the business revenue and potential customer population it can serve with default loss control. This measure could be extended and apply to other financial domains which have strict and specific prediction error limitations as a domain-specific evaluation metric, such as insurance and consumer lending.

For a given risk score threshold $t(p)$ where $p$ is the predefined bad rate, the approval rate is calculated as

$$\text{Approval rate} = \frac{\text{number of applications with score} \leq t(p)}{\text{number of applications}}$$

$$\text{Bad rate } p \text{ from accepted loans} = \frac{\text{number of applications with label Bad}}{\text{number of applications with labels in } \{\text{Good, Bad}\}}$$

Note that the calculation of approval rate is based on both labeled and unlabeled data. Refer to Figure 3 for an illustration of risk score distribution and its relation to approval rate calculation. Given all the applications, predicted risk scores, and a specific risk score threshold, applications whose risk scores are lower than the threshold assume to be approved. And the corresponding bad rate, which is the bad rate of labeled applications, is the ratio of bad labels in the labeled data in the approved application set. The bad rate of the overall approved population will be larger and challenging to estimate because of the lacking of ground-truth labels of rejects. So we use the bad rate of labeled applications as bad rate alternative. To take unlabeled data into consideration, such bad rate thresholds from labeled applications need to be set lower than normal business bad loan rates that financial institutes could take. Therefore, we report approval rate estimates on different low bad rates, including 2.5% and 3.5% in the results.

## Experiment Results

Table 2 summarizes the performances of our proposed self-training method and benchmarks on the test set. The training size reports the final training size applied in each method compared to the original accepted data size. Fuzzy augmentation uses the full unknown data for the training, while Self-training method selectively includes part of unlabeled dataset into the training set. The best of each metric is highlighted. Between the two benchmarks, fuzzy augmentation does not improve the performance compared to KGB model on most of the metrics. Especially for weighted logistic regression, evaluation metrics of fuzzy augmentation drop significantly. One possible explanation is that the training set in fuzzy augmentation shift largely from labeled data for remedying data bias. However, the classification accuracy is compromised.

For Self-training method, XGBoost algorithm outperforms benchmarks, and it works better with calibrated prob-

| Credit risk model | Method[a] | Training size | AUC | K-S | Approval rate[b] @ 2.5% bad rate | Approval rate @ 3.5% bad rate |
|---|---|---|---|---|---|---|
| XGBoost | KGB | 100% | 0.737 | 0.367 | 0.529 | 0.696 |
| | FA | 238% | 0.738 | 0.365 | 0.510 | 0.684 |
| | ST-TS | 126% | 0.732 | 0.356 | 0.514 | 0.704 |
| | ST-CP | 115% | **0.740** | **0.381** | **0.543** | **0.706** |
| Logistic Regression | KGB | 100% | 0.732 | 0.359 | 0.515 | **0.716** |
| | FA | 238% | 0.720 | 0.336 | 0.455 | 0.669 |
| | ST-TS | 117% | **0.733** | **0.362** | **0.521** | 0.708 |
| | ST-CP | 102% | 0.732 | 0.354 | 0.506 | 0.705 |

Table 2: Performance of Self-training Methods and Benchmarks in the Experiment
[a] KGB: Known Good/Bad model, FA: Fuzzy Augmentation, ST-TS: Self-training based on Trust Score, ST-CP: Self-training based on calibrated probability, [b] Approval rate: approval rate estimate based on labeled and unlabeled datasets.

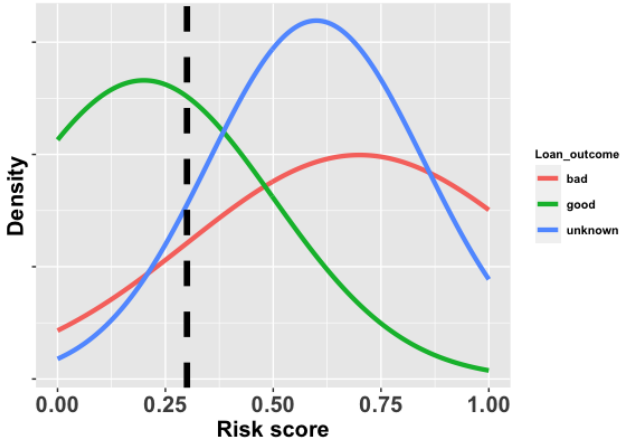


Figure 3: Illustration of approval rate calculation. The dashed line is risk score approval threshold $t(p)$: applications with lower risk scores are approved.

ability as confidence model than Trust Score. Self-training with probability calibration confidence model outperforms all other methods in AUC, K-S statistics, and approval rate @2.5% bad rate. Compared to the KGB model, the approval rate increases from 52.9% to 54.3%, and K-S statistics improves from 0.367 to 0.381. In contrast, self-training with TrustScore only performs better than fuzzy augmentation on the approval rates.

The evaluation of weighted logistic regression implies different results compared to XGBoost. The results of KGB, self-training with Trust Score, and self-training with calibrated probability are close to each other for all metrics. Self-training with Trust Score performs slightly better than the others, K-S statistic of 0.362, AUC of 0.733, approval rate @2.5% bad rate of 0.521, without statistical significance. Logistic regression returns well-calibrated predictions by default as it optimizes log loss. Therefore, probability calibration is not expected to improve the performance, and the training set (102% of original training set) is almost not augmented in the experiment. In general, XGBoost has better classification performance than weighted logistic regression in most metrics and the benefit provided by self-training is more significant on XGBoost algorithm.

Performance gains are relatively modest, consistent with the prior literature (Hand and Henley 1993). Friedman's rank sum test reports that not all the methods perform the same ($p < 0.05$), but pairwise comparisons do not show significant differences between most of the methods. However, considering the large loan volume involved, it is still a significant difference for business purposes.

The final training set sizes among different methods in Table 2 also implies the need for fine-grained selection for inferred training set as discussed in other literature (Li et al. 2020). Introducing more pseudo-labels into training set does not guarantee better classification performance. The introduction of data that are far from decision boundaries of classifiers may not help with classification performance. As prediction uncertainty studies develop, one future work direction is to consider new uncertainty estimates to improve the self-training performance for RI models.

## Model Deployment and Early Results

### Model Production Deployment

Based on the experimental results in Table 2, the self-training method with calibrated probability (ST-CP) and KGB model using XGBoost as credit risk algorithm are chosen and was deployed into production since March 2021. The credit risk scores generated are applied in the loan approval process as reference. Unlike chatbots and recommendation systems, the loan approval policy must stay consistent and fair for all applicants. Therefore, only the KGB model scores are used in the current approve/decline process. ST-CP model scores are trained with previous loans in one batch, monitored in silent mode and only serves as benchmark purpose at the moment. We have been actively monitoring the performances of self-training model and KGB model closely and run regular comparison analyses.

Figure 4 illustrates the abstract architecture design of the model deployment pipeline. First, the loan application starts from one of the access points such as QB Capital website (Capital 2021), email promotion, in-product discovery banner in QBO, direct mail or other internet channels with QB Capital advertisement. After loan application is submitted, internal and external data are fetched and transformed into

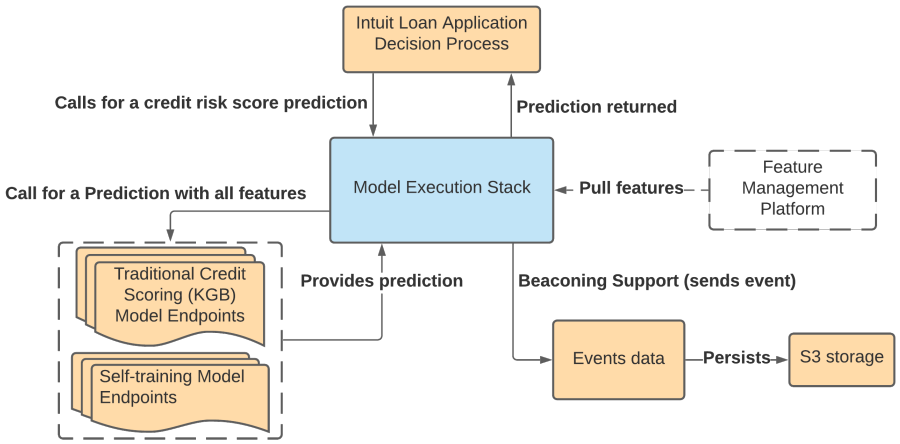


Figure 4: Model deployment architecture

features in real time and an API call is made to model execution stack where trained model endpoints are hosted. The model execution stack composed of two model endpoints, KGB model and Self-training model. Model prediction results are returned to the caller. Logging and monitoring of the model inputs and outputs along with metadata is handled through Apache Kafka streaming events and eventually persisted into designated AWS S3 buckets. Note that model hosting, execution, logging, and monitoring are done on Intuit Machine Learning Platform (Canchi and Wenzel 2020).

## Early Deployment Results

As an evaluation of the performance in the production environment, an anonymous sample of about 10,000 loan applications and a few hundred matured loans were collected and analyzed. The K-S statistic of applications with loan outcomes between models are compared, while approval rate estimates are reported, shown in Table 3. Note that we still report approval rate estimate here since the production application decisions are still based on traditional credit scoring model (KGB model).

These results show that our proposed self-training model outperforms the traditional credit scoring method across the metrics. Self-training method improves the K-S statistics by 6.8%, which indicates that our proposed model performs better in detecting potential default applications. And estimated approval rate increases from 70.7% to 76.9% when bad rate is 3.0%.

We run another evaluation to measure business impact of reject inference compared to current approval policy: we compare the bad rate of observed approval population and predicted approval population from self-training method. For the 76.9% of self-training model -'approved' applications whose bad rate on labeled population is 3%, we predicted the bad rate of all RI-approved applications by assigning a higher bad rate to rejects. Similar to the bad rate that we observed in the matched bureau data for the rejects, we assume rejects (unlabeled) have 3 times higher default risk than observed labeled data, and the result shows that the predicted overall bad rate is 5.1% for both models.

| Model | K-S | Est. Approval rate @ 3% bad rate[a] | Est. population bad rate [b] |
|---|---|---|---|
| KGB | 0.441 | 0.707 | 5.1% |
| ST-CP | **0.471** | **0.769** | 5.1% |

Table 3: Performance Comparison of Self-training and KGB models in Production

[a] 3% bad rate is based on labeled population; [b] Assume loan default risk is 3 times higher in the rejects.

On the other hand, among all loan applications in this sample, 55% were approved by the current lending policy, and the observed bad rate is 5.2%. With similar bad rates on approved population, these results prove that Self-training method largely increases the approved population lending institutions can serve by 39% while keeping the same low default risk.

## Conclusion and Future Work

In this paper, we have shared application of self-training methods that can help reducing sampling bias in credit risk models and how we should evaluate these methods based not only on the traditional model performance metric but also on business KPI related metric, i.e., application approval rate.

We had empirically shown in both historical loans and production that including data selectively from the loan applications with unknown outcome with self-training approach can effectively improve credit risk models in terms of their performance on the general population, i.e., approving more applications while maintaining the same risk level. We are cautiously optimistic about the early production results and will continuously monitor the proposed model performance in the next few months. This warrants the consideration of making proposed model "official" for the sake of making better lending decisions and further research to explore more ML related sampling bias correction methods in the credit scoring domain and other financial fields.

## References


Amazon-Sagemaker. 2020. How Hyperparameter Tuning Works. https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning-how-it-works. Accessed: 2020-07-01.

Bradley, A. P. 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7): 1145–1159.

Canchi, S.; and Wenzel, T. 2020. Managing ML Models @ Scale - Intuit's ML Platform. USENIX Association.

Capital, Q. 2021. Get business funding when it matters. https://quickbooks.intuit.com/capital/.

DMLC/xgboost. 2016. [New Feature] Monotonic Constraints in Tree Construction. https://github.com/dmlc/xgboost/issues/1514.

Ehrhardt, A.; Biernacki, C.; Vandewalle, V.; Heinrich, P.; and Beben, S. 2021. Reject inference methods in credit scoring. *Journal of Applied Statistics*, 1–21.

Eisenbeis, R. A. 1977. Pitfalls in the application of discriminant analysis in business, finance, and economics. *The Journal of Finance*, 32(3): 875–900.

Feelders, A. 2000. Credit scoring and reject inference with mixture models. *Intelligent Systems in Accounting, Finance & Management*, 9(1): 1–8.

Hand, D. J.; and Henley, W. E. 1993. Can reject inference ever work? *IMA Journal of Management Mathematics*, 5(1): 45–55.

Hsia, D. C. 1978. Credit scoring and the equal credit opportunity act. *Hastings LJ*, 30: 371.

Jiang, H.; Kim, B.; Guan, M.; and Gupta, M. 2018. To trust or not to trust a classifier. In *Advances in neural information processing systems*, 5541–5552.

Kolmogorov, A. 1933. Sulla determinazione empirica di una legge di distribuzione. *G. Ist. Ital. Attuari.*, 4: 83–91.

Kozodoi, N.; Katsas, P.; Lessmann, S.; Moreira-Matias, L.; and Papakonstantinou, K. 2019. Shallow Self-Learning for Reject Inference in Credit Scoring. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 516–532. Springer.

Li, Z.; Hu, X.; Li, K.; Zhou, F.; and Shen, F. 2020. Inferring the outcomes of rejected loans: an application of semisupervised clustering. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*.

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

Lundberg, S. M.; and Lee, S.-I. 2017. A Unified Approach to Interpreting Model Predictions. In *AProceedings of the 31st international conference on neural information processing systems*, 4768–4777.

Montrichard, D. 2007. Reject inference methodologies in credit risk modeling. *SESUG 2008*.

Nguyen, H.-T.; et al. 2016. Reject inference in application scorecards: evidence from France. Technical report, University of Paris Nanterre, EconomiX.

Niculescu-Mizil, A.; and Caruana, R. 2005. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, 625–632.

Siddiqi, N. 2003. *Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring*. Princeton University Press. ISBN ISBN 0-691-09046-7.

Smirnov, N. 1948. Table for Estimating the Goodness of Fit of Empirical Distributions. *The Annals of Mathematical Statistics*, 19(2): 279–281.

Triguero, I.; García, S.; and Herrera, F. 2015. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information systems*, 42(2): 245–284.

Wang, W.; Lesner, C.; Ran, A.; Rukonic, M.; Xue, J.; and Shiu, E. 2020. Using Small Business Banking Data for Explainable Credit Risk Scoring. In *The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, New York, NY, USA, February 7-12, 2020*, 13396–13401. AAAI Press.