Statistical Learning for Credit Risk Modelling

Veronica Bacino, Alessio Zoccarato, Caterina Liberati and Matteo Borrotti
University of Milano-Bicocca

Introduction

Objective. Develop a quantitative Credit Scoring (CS) model that can distinguish between good and bad applicants [2]. In particularly the CS model will estimate the probability that an applicant will be able to pay off the debit taken out with the bank. The CS model will be primarily based on financial/economic ratios computed on the client banking account.

State-of-the-arts. A variety of techniques have been applied in such predictive learning problem [2, 6, 8, 9, 10] with different success. Xiaaet al. (2017) [13] pointed out that ensemble classifiers perform better compared to single classifiers.This is also justify in accordance with the "no free lunch theorem" [12]. One issue of CS models is related to the ratio between good users and bad users, which leads to a severe data imbalance ratio. Imbalanced datasets come with certain challenges for the construction of a classification model. One commonly approach is to oversampling or undersampling the target variable [7]. Another issue is related to the number of hyper-parameters that should be tuned on recent ensemble algorithms [13].

Contribution. We investigate the performance of Bayesian Optimization (BO) [1] and eXtreme Gradient Boosting (XGBoost) algorithm [3] together with a cost sensitive learning approach [4] for imbalanced data for developing a credit scoring model.

References

[1] Archetti, K.O., Candelieri, A.: Bayesian Optimization and Data Science. Springer (2019).

[2] Baesens, B., Van Gestel, T., Viaene, S., Stepanova, M., Suykens, J., Vanthienen, J.: Benchmarking state- of-the-art classification algorithms for credit scoring. J. Oper. Res. Soc. **54**, 627—635 (2003).

[3] Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. arXiv arXiv:1603.02754, 1–13 (2016).

[4] Elkan, C.: The Foundations of cost-sensitive learning. In: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, pp. 973–978 (2001).

[5] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., Schmidhuber, J.: LSTM: A Search Space Odyssey. IEEE Trans. Neural Networks Learn. Syst. **28**(10), 2222—2232 (2017).

[6] Hand, D. J., Henley, W. E.: Statistical classification methods in consumer credit scoring. J. R. Stat. Soc. **160**(3), 523–541 (1997).

[7] He, H., Zhang, W., Zhang, S.: A novel ensemble method for credit scoring: Adaption of different imbalance ratios. Expert Syst. Appl. **98**(2018), 105—117 (2018).

[8] Huang, C. L., Chen, M. C., Wang, C. J.: Credit scoring with a data mining approach based on support vector machines. Expert Syst. Appl. **33**(4), 847—856 (2007).

[9] Li, X., Ying, W., Tuo, J., Li, B.: Applications of classification trees to consumer credit scoring methods in commercial banks. In: Proccedings of IEEE international conference on systems, man and cybernetics, pp. 4112—4117. IEEE, New Jersey (2004).

[10] West, D.: Neural network credit scoring models. Comp. Oper. Res., **27**(11), 1131–1152 (2000).

[11] Williams, C.K.I. and Rasmussen, C.E.: Gaussian processes for machine learning. MIT Press (2006).

[12] Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput., **1**(1), 67–82 (1997).

[13] Xiaa, Y., Liua, C., Lib, Y., Liua, N.: A boosted decision tree approach using Bayesian hyper-parameter optimization for credit scoring, Expert Syst. Appl. **78**(2017), 225—241 (2017).

Acknowledgements

We greatly acknowledge the DEMS Data Science Lab for supporting this work by providing computational resources.

datalab
BICOCCA DATA SCIENCE LAB

Methods

eXtreme Gradient Boosting (XGBoost). XGBoost algorithm was recently proposed by Chen et al. (2016) [3]. XGBoost optimizes the objective function and its estimation. The fast, efficient, and scalable system achieves promising results on numerous standard classification benchmarks. XGBoost combines a series of weak base learners, which are normally regression trees, into a strong one. The weak learner herein refers to a model that only performs slightly better than a random guess. Boosting fits additive base learners to minimize the loss function provided. Loss function measures how well the model fits the current data. The process of boosting continues until the loss function reduction becomes limited. For a more detailed description see Chen et al. (2016) [3].

Bayesian Optimization (BO). BO is a sample-efficient strategy for global optimization of black-box, expensive and multi-extremal functions, traditionally constrained to over a box-bounded search space Ω:

$$\min_{\theta \in \Omega} g(\theta)$$

BO is base on two key components: a *probabilistic surrogate model* (*i.e.* Gaussian Process [11]) of the objective function $g(\theta)$ in order to provide an estimate of $g(\theta), \forall \theta \in \Omega$, along with a measure of uncertainty about such an estimate and an *acquisition function* that is based on the current approximation of $g(\theta)$. The optimization of the acquisition function allows to select the next promising θ' where to evaluate the objective function. The observed value, $g(\theta')$ (or $g(\theta') + \varepsilon$ in the case that the objective function is also *noisy*), is then used to update the probabilistic model approximating $g(\theta)$, and the process is iterated until a given termination criteria is reached (*e.g.*, a maximum number of function evaluations). One of the most widely used acquisition functions is Upper Confidence Bound (UCB) that manages exploration—exploitation by being optimistic in the face of uncertainty. Several acquisition functions have proposed - an overview is provided in Archetti et al. (2019) [1] - each one offering a different mechanism to balance the exploitation-exploration trade-off.

Cost-sensitive learning. Classifiers are designed to minimize the number of errors (incorrect classifications) made. When misclassification costs vary between classes as in credit scoring, this approach is not suitable. A possible solution is to balance the classes according to their costs re-weighting the training examples in proportion to their costs [4].

Data

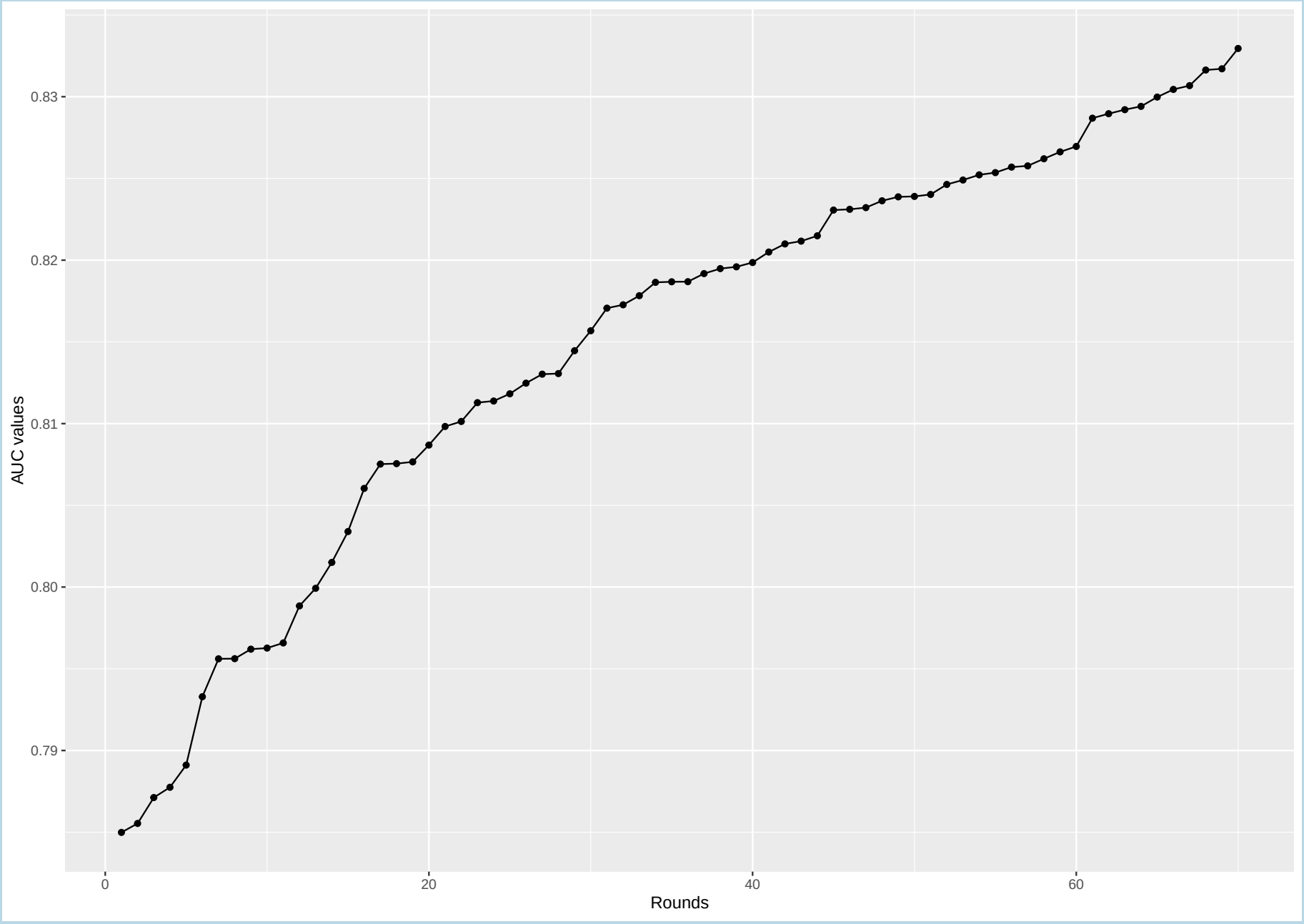
The dataset is composed by 7500 individuals that applied for a bank loan since June 2015 until February 2020.

Target variable. According to the consumer credit regulation, we computed our target variable (Y) as a dummy, checking the clients behaviour at the end of 12 months after the loan acquisition. Specifically, we labelled an applicant not creditworthy (Y=1) if she/he had at least three installments to repay still, otherwise we labelled her/him creditworthy (Y=0).

Input variables. The input variables of our model are 83 and have different metrics: they are dummy (28), counting (16), numerical (39). They have been computed in order to investigate different aspects of the financial behaviour of the customers. More in detail, 27 variables are related to the capacity of the client to have positive cash flow (Capacity), 23 to the client reliability (Reliability), 13 to the variety of banking payments different from cash (Bank intensity), 2 to presence of life insurances (Protection seek) and 16 related to the planning behavior respects to the expenditures (Lending behavior).

Results

On our credit risk model, BO is used to optimize a cost sensitive learning version of the XGBoost algorithm, from now on BO_costXGBoost, on which the balance of creditworthy and not creditworthy is adjust by a specific weight sets as the ratio between the two class labels.



Hyperparameter	Values
eta	{0.01, 0.05, 0.1, 0.3}
max.depth	{1, 3, 5, 7}
min_child_weight	{1, 3, 5, 7}
subsample	{0.5, 0.8, 1 }

The best configuration is eta = 0.05, max.depth = 2, min_child_weight = 3 and subsample = 0.51 and it reaches an AUC value of 0.833.

Two approaches are compared against the proposed solution: a standard version of the XGBoost (dafaultXGBoost) and a XGBoost with a cost sensitive learning approach (costXGBoost).

	Recall	**F1-score**	**Type II error**	**Accuracy**
defaultXGBoost	0.105	**0.178**	0.895	**0.975**
costXGBoost	0.105	0.131	0.895	0.965
BO_costXGBoost	**0.605**	0.163	**0.395**	0.843