

Predictive model optimization by searching parameter space

Grzegorz Harańczyk^{1,2}

¹ StatSoft Polska,
Kraszewskiego 36, 30-110 Krakow, Poland

² Institute of Mathematics, Jagiellonian University,
Łojasiewicza 6, 30-348 Krakow, Poland
g.haranczyk@statsoft.pl

Abstract. The process of model building involves making a series of decisions. In practice the best values for parameters of analysis depend on the problem, and they should be carefully attuned. In this article we present a proposition how to optimize a predictive model by searching the parameter space.

Keywords: unbalanced class problem, random forest, boosting trees, misclassification error, AUC.

1 Introduction

In this article we present our approach to the problem of prediction client delinquency. We present the essential stages of the process, focusing on the aspects which make the solution innovative. The problem of prediction client delinquency is an example of classification problems. Datasets consist of 52 explanatory variables of several types. Our goal was to find a predictive relationship between the inputs and target variable with values 1 for bad clients and 0 for good clients.

1.1 Main issues and the essential features of the innovative approach

Main issues were binary target variable with unequal class sizes, not very good quality of data and model re-calibration problem. The another problem was that some predictor variables with many categories - too many to look for rules (e.g., variable RESIDENCIAL_BOROUGH). In this case we were looking for similarities. We used text mining approach to retrieve information from multiple categorical predictor variables. Model performance was evaluated by AUC so we were searching parameter space to find optimal predictive model with respect to this criterion. We used some numerical simulation to obtain the best parameters of analysis.

2 Data Preparation

1.1 Missing data and redundant variables

Missing data and values of type 'XX' etc. were recoded and the special categories were created. Some of the variables were constant (CLERK_TYPE) or the same as the other ones (QUANT_SPECIAL_BANKING_ACCOUNTS was the same as QUANT_BANKING_ACCOUNTS). These variables were excluded from subsequent analyses. Some variables have values only for leaderboard and prediction datasets (e.g., FLAG MOBILE PHONE). We can not make any rule depending on this variables. These variables were also excluded from subsequent analyses.

1.2 New variables and text mining

There was a problem with multi-categorical predictor variables. For example, there was too much unique codes for predictor variables: RESIDENCIAL_BOROUGH, PROFESSIONAL_BOROUGH. We performed a text mining analysis and added dummy variables coding word occurrence in borough name (e.g. alto, centro, nova, novo, rural, santa, santo, sao, vila, vista, zona).

Additional information about Brazilian regions was merged with dataset. It helped to handle with other predictor variables with many unique categories e.g., CITY_OF_BIRTH and PROFESSIONAL_CITY. These variables were replaced by population of these cities. Moreover, additional information about states of Basil was used: area, population in 2005, density in 2005, GDP, GDP per capita PPP, Human Development Index (HDI), literacy rate, infant mortality, life expectancy (based on [10]).

Also some new variables were computed (some summary scores or logical combinations of two or more variables).

1.3 Variable selection

We obtained a large set of predictor variables that were related to the outcome of interest. In the next step we wanted to obtain the smallest possible set of variables without essential loss of information. We used a random forest algorithm (similar approach was presented in [8]).

2 Modeling and evaluation

Two types of classification algorithms were used to build final models: boosted trees (stochastic gradient boosting trees) [1,4,5] and random forest [2,7]. In both cases we used a *STATISTICA* implementation [9]. In presented problem, a criterion for a model performance was the area under an ROC curve (AUC). However, the objective function optimized in most of algorithms is the error rate and not the AUC value. Relationship between the AUC and the error rate was presented for example in [3]. We made some numerical simulation to obtain the best analysis parameters for AUC-performance.

Class of bad clients was under-represented (26% in modeling dataset). Additionally, challenging was the fact that it was expected that the delinquency rate increases in the *Prediction dataset*, once that it also contains "rejected clients".

Despite class imbalance we wanted to ensure that the classifier does not only predict the majority class. There are quite a few approaches to the problem (e.g., cost-sensitive learning, up-sampling, and down-sampling). We chose cost-sensitive learning, i.e., we wanted to set different misclassification costs for different classes. In the implementation of the stochastic gradient boosting trees algorithm in *STATISTICA*, the prediction probabilities for each class are adjusted for unequal prior probabilities and misclassification costs [1]. However, in case of a random forest algorithm, we changed a misclassification costs. The best value was chosen based on simulations. A common approach is to set misclassification costs inversely proportional to the class populations. The weight of 3 on class of delinquents was too high. The best value was chosen according to numerical simulations (Fig. 1). By the way, we were looking for the complexity of the optimal model.

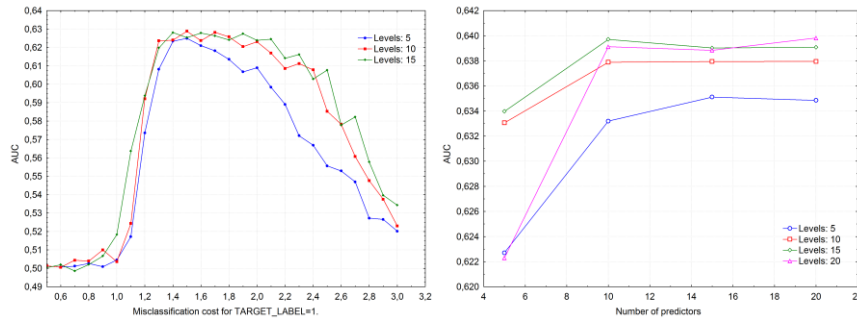


Fig. 1. Relationship between parameter of analysis (right: misclassification error rate, left: number of predictors) and model performance for a random forest model.

Next step was to tune other parameters of models. They were also evaluated by numerical simulations.

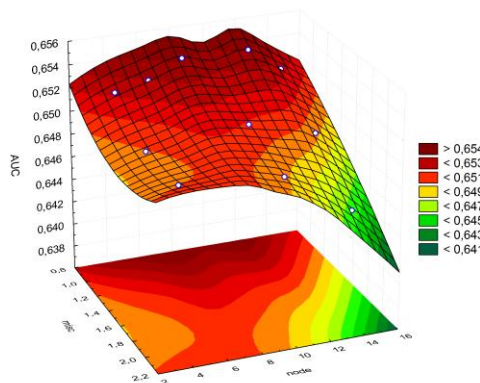


Fig. 2. Relationship between parameter of analysis (misclassification error rate and complexity of trees) and model performance for boosting trees model.

For a random forest method the parameter space was three dimensional – the number of trees, number of predictors, complexity of trees (number of levels). The number of predictors for the tree models (number of variables randomly sampled as candidates at each split) is recommended to be square root of a number of all variables [6]. This was consistent with our simulations (Fig. 1).

For a boosting trees we were considering the number of trees, complexity of trees (criterion based on maximum of nodes) and also misclassification costs were explored. Medium complexity of trees turned out to be optimal (Fig. 2). It can be easily observed, as it was mentioned, that there is no need to tune the problem with imbalanced classes. The last parameter - the number of trees, was chosen during the process of learning (we used 10-fold cross validation).

During model building there was no special assumption that in *Prediction dataset* there will be more delinquents but such a tendency can be observed in the obtained scoring results (Fig. 3).

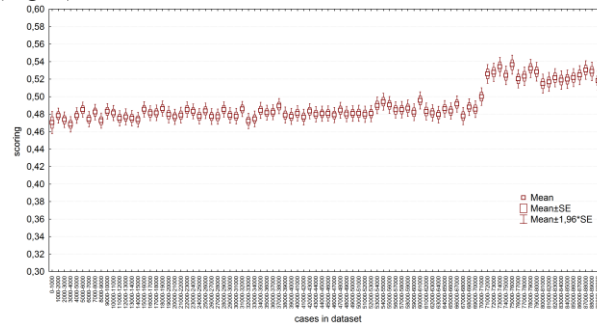


Fig. 3. The mean scoring for successive cases in dataset grouped by 1000 cases.

References

1. Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J.: Classification and regression trees. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software, (1984)
2. Breiman, L.: Random Forests, Machine Learning, vol. 45 (1):pp. 5–32, (2001)
3. Cortes, C., Mohri M.: AUC Optimization vs. Error Rate Minimization. In: Thrun, S., Saul, L. K., Schölkopf, B. (eds.) Advances in neural information processing systems 16: proceedings of the 2003 conference, pp. 313-320. MIT Press, (2004)
4. Friedman, J. H.: Greedy function approximation - a gradient boosting machine, Annals of Statistics, vol. 29: pp. 1189-1232 (2001)
5. Friedman, J. H.: Stochastic gradient boosting, Comput. Stat. Data Anal. 38, 367–378, (2002)
6. Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning, Springer, (2008)
7. Ho, T. K.: Random Decision Forest, 3rd Int'l Conf. on Document Analysis and Recognition. pp. 278–282, (1995)
8. Sandri, M., Zuccolotto P.: Variable selection using random forests, Data Analysis, Classification and the Forward Search. Part IV, 263-270, Springer, (2006)
9. StatSoft, Inc. (2010). STATISTICA (data analysis software system), version 9.1. www.statsoft.com.
10. http://en.wikipedia.org/wiki/States_of_Brazil