Enhancing Predictive Accuracy in Large Datasets with High Missingness: An XGBoost Approach with Bootstrap Resampling

Xiaolong Wang, Zhouchi Ni, Chenxing Liao

2023-12-13

Introduction

In the evolving landscape of multilevel prediction, the reliability and accuracy of predictive models have become paramount. This report explored the application of bootstrap resampling methods within the framework of an XGBoost predictive model, utilizing a substantially large dataset encompassing more than 17,000 samples and 1,206 variables collected from a periodical survey from 4000 individuals about their personal situations and health status between 2005 and 2019. A unique challenge presented in our dataset is the high incidence of missing values, exceeding 70% in numerous instances.

The bootstrap method, a resampling technique widely recognized for its efficacy in estimating the accuracy and stability of prediction models[1]. In the context of training a model like XGBoost, it is also a method used to improve the robustness and performance of machine learning algorithms. Bootstrapping offers several benefits like reducing overfitting, estimating model accuracy, quantify uncertainty[2], etc. Our investigation primarily focuses on assessing the influence of varying the bootstrap sample size and the number of iterations on the predictive accuracy of the model.

The choice of sample size in bootstrap resampling is critical, as it directly impacts the ability of the model to capture and reflect the internal statistical properties of the original dataset. Moreover, the number of bootstrap iterations also plays a crucial role in balancing the trade-off between computational efficiency and the reduction of random sampling errors, which are instrumental in enhancing the model's predictive performance. By navigating these two critical aspects of bootstrap resampling—sample size and iteration count, this report aims to provide insights into optimizing the performance of complex machine learning models in the face of large, incomplete datasets.

Data

This comprehensive dataset consists of the responses of 4,000 individuals surveyed periodically between 2005 and 2019. The survey contains a wide array of questions that delve into various aspects of the participants' lives, including their financial circumstances, political affiliations, and familial dynamics, such as the number of children in each household. Each participant's record is uniquely identified by a 'personid', with the survey year also noted for longitudinal analysis. One of the critical elements of this dataset is the self-assessment of health status, where respondents rated their health on a scale from 1 (Very Good) to 5 (Bad). The dataset contains over 1200 variables (labelled from x1 to x1205, plus 'health'), providing opportunities for investigations on a diverse range of topics. This rich, longitudinal data provides great chances to explore trends and patterns in life experiences and health status and practice data-cleaning techniques and machine learning algorithms.

Method and Implementation

Bootstrapping

The fundamental principle behind bootstrapping is to mimic the process of obtaining new sample data by resampling from the existing data, allowing for the calculation of various statistics, like means, variances, and confidence intervals. This approach is particularly useful in situations where the theoretical distribution of an estimator is complex or unknown. In the context of predictive modeling, by repeatedly resampling the dataset and recalculating the model, we can evaluate the variability and bias of the prediction.[3]

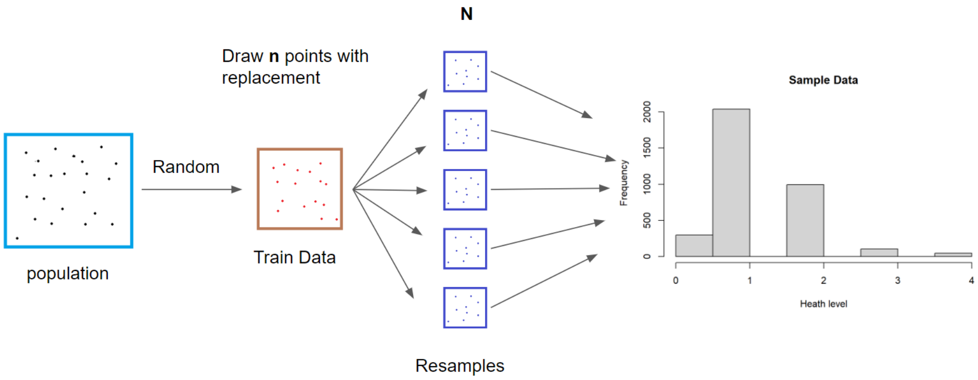

Figure 1: Bootstrapping

XGBoost

XGBoost[4] (eXtreme Gradient Boosting) is an advanced implementation of gradient boosting algorithms. XGBoost is an ensemble learning method that constructs a series of decision trees in a sequential manner, where each subsequent tree aims to correct the errors made by the previous ones.

In case of handling missing data, XGBoost handles missing data by learning the best direction to split for each feature with missing values during the tree construction process. Instead of imputing missing values, it decides whether to send them left or right in a tree branch based on which choice optimizes the model's performance. This approach is applied both during training and prediction, allowing XGBoost to deal with missing data directly without the need for preliminary imputation.

Double Bootstrap

Since we chose to use the mode as our attribute of prediction, to estimate the variance we need to introduce some advance techniques due to the mode's less stable distribution. Thus, in this case, we decide to implement double bootstrap.[5] The double bootstrap algorithm enhances variance estimation through a two-tiered resampling process. Initially, it generates multiple bootstrap samples from the original data. For each of these samples, a second round of bootstrapping is performed to create sub-samples. The nested resampling allows for a more accurate variance estimation by simulating the distribution of the mode in the first resampling.

Method

The aim of our study is to investigate the efficiency of bootstrapping method on missing data in a data set. Therefore, the first part of our project was dividing the the data set to a train data set and a test data set which could give a gold standard to calculate the accuracy of the predict value from our model. Then we used bootstrapping method on the train data to obtain N resampled data set and set the size of each resample to n. In order to find the proper value of N, we input different number in N and determine the performance the accuracy of each number of N in our model.

To be more specifically, we chose 1, 10, 100, 1000, 2000 as possible most effective iteration numbers of bootstrapping in this train data set. For each number we chosen, R code has been written to operate bootstrapping process. After getting N resamples, we fit a gradient boosted decision trees (XGBoost) model and using the package Xgboost in R to get its prediction for the test data set as a list writing in a .Rdata file. Since the XGBoost method could deal with missing value by itself. We set hyper parameters, such as a learning rate of 0.1 and 100 boosting rounds. This parameter set will make the model more robust since it includes a small learning rate and a large boosting round. We set the maximum tree depth as 6. These choices were guided by the need for a balance between model complexity and generalization.

Since the number N includes 100 to 2000, this would take a large amount of time to produce all Rdata file by a personal computer. To make this iteration process more efficient, we used the GreatLake cluster to seamlessly execute those iterations. Each resample and model building would cost one to two minutes in our own laptop and takes more than 24 hours if we set N equal to 2000, while the cluster computer which runs all the jobs in parallel could deal with the whole project around 10 minutes. We wrote a batch file and upload with R script and data sets and run the batch file. The output predicting R data sets were store on the GreatLake and easy to download to our own computer for final processing.We set the parameter in the GreatLake cluster by the batch file. The estimate time to complete the whole process was 300 hours, the size of memory was 1 GB and we only used one CPU for each job since the character of R. The array value was equal to the iteration number N and the partition was standard. In terms of the module in this part, Rtidyverse was loaded to operate the R script.

Then, we could get a summary matrix containing predictor value from all N resamples by loading all the Rdata file. The last part was using the summary matrix to get a final predict data for the test data set. This would be a part of bootstrapping that includes several methods to analysis the data. In our project, we chose the mode of each column in the data set which represent every test sample's final health level with its estimated error interval. For the interval of prediction, we used double bootstrapping method. We got an extra bootstrapping step for each resample data set with 100 iteration number and 20000 samples as sample size. Then we found the mode of every 100 resample data set in this step. After that part, we store those mode data in a matrix with 3480 column and find variance for each column. Using the variance of each predict value, we could calculate the estimated error interval.

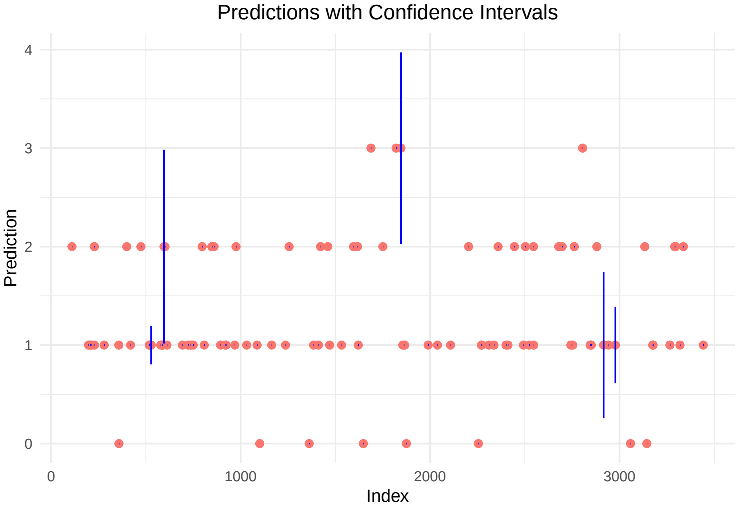

Figure 2: sample with interval

We used two list to store the upper boundary and the lower boundary of the predictor value and compared with the health level in test data. If the actual healthy level states inside the interval, we would treat the predict value as a correct one. Finally, we could count all correct value and calculate the proportion of correct value as the accuracy of our model.

After the whole project we could find the most accurate iteration number for our bootstrapping model. The second part of the project was to study the performance of the figure of resample sizes n. In the first part, we set each resample size to 10000 data points. Then we set n to 1000, 2000, 5000, 10000, 20000 and found the accuracy of each model. The whole process was similar to the first part including drawing n points to be new resamples, building XGboost models, getting summary matrix and comparing with the test data.

Conclusion

Using these data above, different sample size and iteration number will result in different accuracy. When the sample size is n, n observations were selected with replacement, which means the number of observations that were selected for fitting one model is n. When the iteration number is n, it means there will be n models for preditions and each model can provide one prediction.

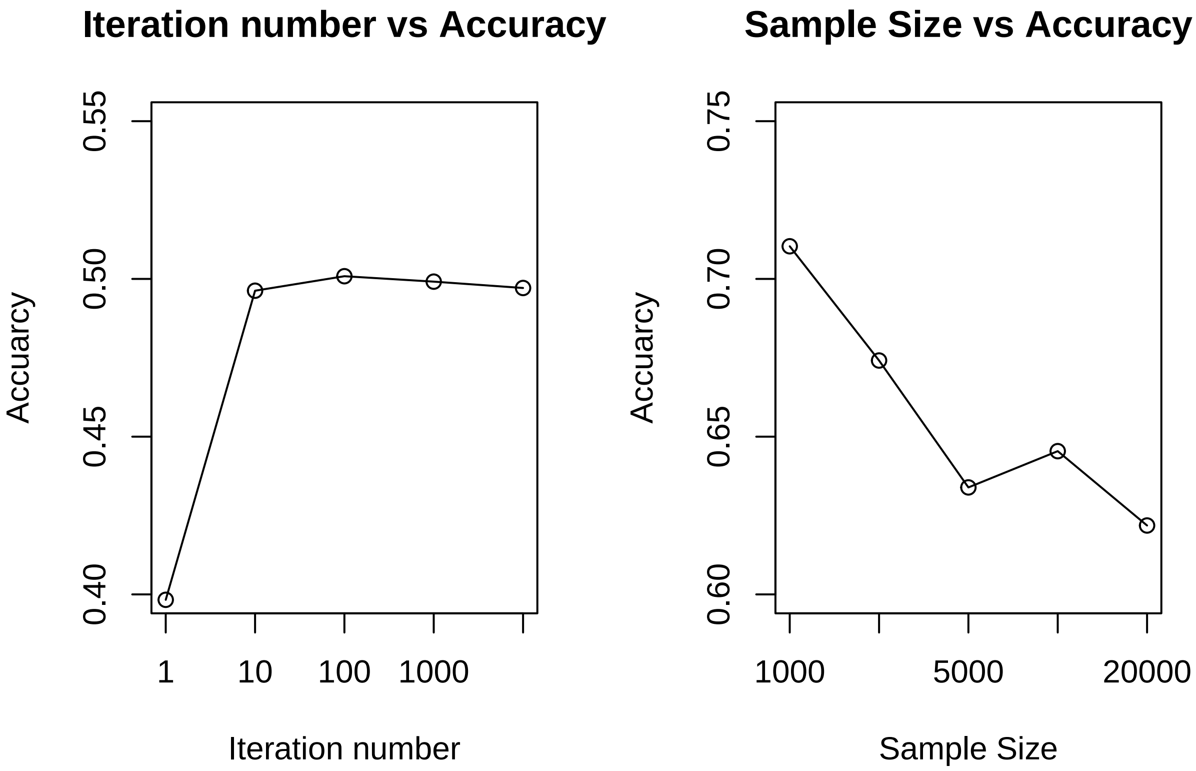

Figure 3: Iteration number and Sample Size vs Accuracy

The prediction basis is mode, the values that appear most frequently will be predicted values. Comparing the accuracy can help determine the sample size and iteration number. Five different iteration numbers are tried when sample size is fixed as 10000. According to the plot, the accuracy is not always better when iteration number gets greater. From the plot the best iteration number for accuracy is 100. Since the best iteration number is 100, another plot can show the association between sample size and accuracy when the iteration number is fixed as 100. According to this plot, in the situation that sample size is below 20000, the greater the sample size, the worse the accuracy is.

Discussion

According to the plot of Sample size vs Accuracy. As mentioned in introduction, any trend can be possible, but the main reason that cause this can be the quality of data. The data of interest contains so many missing values. Apart from the missing values, the data that successfully collected might also have some quality problems. As so many rounds of Bootstrapping we did, overfitting is also an issue that cannot be ignored. The trend of Sample size vs Accuracy is not strictly negative, which means this trend might be affected by small change of data or statistical analysis. Statistical analysis is also not always stable.

References

[1]. Borra, S., & Di Ciaccio, A. (2010). Measuring the prediction error. A comparison of cross-validation, bootstrap and covariance penalty methods. Computational Statistics &amp; Data Analysis, 54(12), 2976–2989. https://doi.org/10.1016/j.csda.2010.03.004

[2]. Robert A. Stine (1985) Bootstrap Prediction Intervals for Regression, Journal of the American Statistical Association, 80:392, 1026-1031, DOI: 10.1080/01621459.1985.10478220

[3]. Efron, B., & Tibshirani, R. J. (1994). An Introduction to the Bootstrap. Chapman & Hall/CRC.

[4]. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16).

[5]. Letson, D. and McCullough, B.D. (1998), Better Confidence Intervals: The Double Bootstrap with No Pivot. American Journal of Agricultural Economics, 80: 552-559. https://doi.org/10.2307/1244557

Contribution

Xiaolong Wang - Contribute to most of the early-version codes for the bootstrap and the XGBoost training. Contribute to Introduction, Data, and the subsections in Method including Bootstrapping, XGBoost, and Double Bootstrap.

Zhouchi Ni - Contribute to coding in Greatlake and visualization about the result of this project. Contribute to Method part and plots in the report.

Chenxing Liao - Contribute to prediction analysis using different sample size and iteration number from above. Contribute to Conclusion part.