Predicting borrowers' chance of defaulting on credit loans

Junjie Liang
(junjie87@stanford.edu)

Abstract

Credit score prediction is of great interests to banks as the outcome of the prediction algorithm is used to determine if borrowers are likely to default on their loans. This in turn affects whether the loan is approved. In this report I describe an approach to performing credit score prediction using random forests. The dataset was provided by www.kaggle.com, as part of a contest "Give me some credit". My model based on random forests was able to make rather good predictions on the probability of a loan becoming delinquent. I was able to get an AUC score of 0.867262, placing me at position 122 in the contest.

1 Introduction

Banks often rely on credit prediction models to determine whether to approve a loan request. To a bank, a good prediction model is necessary so that the bank can provide as much credit as possible without exceeding a risk threshold. For this project, I took part in a competition hosted by Kaggle where a labelled training dataset of 150,000 anonymous borrowers is provided, and contestants are supposed to label another training set of 100,000 borrowers by assigning probabilities to each borrower on their chance of defaulting on their loans in two years. The list of features given for each borrower is described in Table 1.

Variable Name	Description	Type
SeriousDlqin2yrs	**Person experienced 90 days past due delinquency or worse**	**Y/N**
RevolvingUtilizationOfUnsecuredLines	Total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits	percentage
Age	Age of borrower in years	integer
NumberOfTime30-59DaysPastDueNotWorse	Number of times borrower has been 30-59 days past due but no worse in the last 2 years.	integer
DebtRatio	Monthly debt payments, alimony,living costs divided by monthy gross income	percentage
MonthlyIncome	Monthly income	real
NumberOfOpenCreditLinesAndLoans	Number of Open loans (installment like car loan or mortgage) and Lines of credit (e.g. credit cards)	integer
NumberOfTimes90DaysLate	Number of times borrower has been 90 days or more past due.	integer
NumberRealEstateLoansOrLines	Number of mortgage and real estate loans including home equity lines of credit	integer
NumberOfTime60-89DaysPastDueNotWorse	Number of times borrower has been 60-89 days past due but no worse in the last 2 years.	integer
NumberOfDependents	Number of dependents in family excluding themselves (spouse, children etc.)	integer

Table 1: List of features provided by Kaggle

2 Method

Random forests is a popular ensemble method invented by Breiman and Cutler. It was chosen for this contest because of the many advantages it offers. Random forests can run efficiently on large databases, and by its ensemble nature, does not require much supervised feature selection to work well. Most importantly, it supports not just classification, but regression outputs as well. However, there are still performance parameters that need to be tuned to improve the performance of the random forest.

2.1 Data Imputation

The data provided by Kaggle were anonymous data taken from a real-world source and hence, it is expected that the input contains errors. From observation, I determined that there were three main types of data that required imputation:

1. *Errors from user input*: These errors probably came from a typo during data entry. For example, some of the borrowers' ages were listed as 0 in the dataset, suggesting that the values might have been entered wrongly.
2. *Coded values*: Some of the quantitative values within the dataset were actually coded values that had qualitative meanings. For example, under the column `NumberOfTime30-59DaysPastDueNotWorse`, a value of 96 represents "Others", while a value of 98 represented "Refused to say". These values need to be replaced so that their large quantitative values do not skew the entire dataset.
3. *Missing values*: Lastly, some entries in the dataset were simply listed as "NA", so there is a need to fill in these values before running the prediction. In particular, "NA" values were found for features `NumberRealEstateLoansOrLines` and `NumberOfDependents`.

Since the choice of method of data imputation could significantly affect the outcome of the prediction algorithm, I tested several ways of doing the imputation:

1. *Just leave the data alone*: This method can only be used for errors in the first 2 categories. Since the random forest algorithm works by finding appropriate splits in the data, it will not be adversely affected even if there were coded keys with large quantitative values in the input. However, this would not work for filling in the "NA" entries in the data, so for this method, I removed the features `NumberRealEstateLoansOrLines` and `NumberOfDependents` completely from the dataset.
2. *Substitute with the median value*: Another way to handle the missing data is to use the median values among all valid data within the feature vector. In a way, this is using an available sample within the dataset that would not affect predictions greatly to fill in the missing entry.
3. *Coded value of -1*: Since the randomForest library that I was using would not take "NA" for an input entry, I simply replaced them with a value of -1 (essentially, a value that did not appear anywhere else) to ensure that "NA" is seen as a separate value in the data.

As will be described below, from tests it was determined that method 3 (filling in missing entries with -1) worked the best in giving prediction accuracy.

Lastly, the labels in the dataset were skewed so that there were about 14 times more non-defaulters than defaulters. This is to be expected since most people do not default on their loans, or try not to. However, this will tend to skew the predictions made by the random forest as well. To overcome this problem, I ``rebalanced'' the input by repeating each row with defaulters 14 times, so that overall, the input file contained an even number of defaulters and non-defaulters.

2.2 Parameter Tuning

One of the nice features of random forests is that there are relatively few parameters that need to be tuned. In particular, the parameters that I tested for optimizing were:

1. *Sample size*: Size of sample to draw at each iteration of the split when building the random forests.
2. *Number of trees*: The number of trees to grow. This value cannot be too small, to ensure that every input row gets predicted at least a few times. In theory, setting this value to a large number will not hurt, as the random forests should converge. However, in my tests it appears that a value that is too large reduces the accuracy of the prediction, possibly due to overfitting.

2.3 Experimental Setup

For the test setup, I set up a data pipeline that first took the raw input and passed it through the data imputation stage. Thereafter, the parameters to the random forests are chosen, and a 4-fold cross validation is done on the labeled inputs. An AUC score is calculated for each of the 4 runs, and the average AUC is used to get an estimate of the performance of the algorithm on the actual test data. The experimental setup is shown in Figure 1.

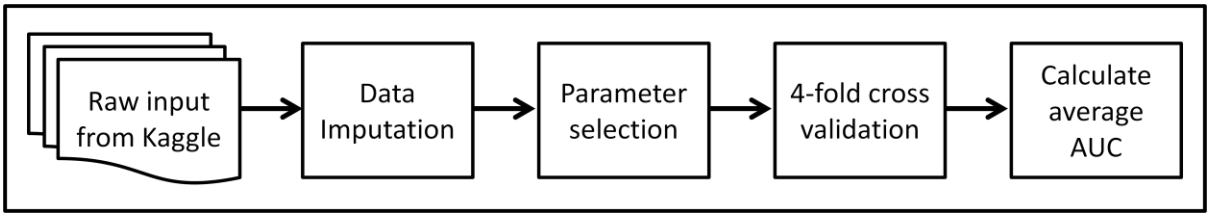

Figure 1: Experimental workflow

3 Results

3.1 Methods of data imputation

I shall now present some of the findings from my experiments. Among the three methods of doing data imputation, it was clear that the method which used coded values gave the best results. This could be because borrowers who had “NA” in their entries were grouped together and used to predict each other’s credit reliability, which would implicitly mean a “nearest neighbor”-like match was being done. The results of the three methods can be seen in Figure 2.

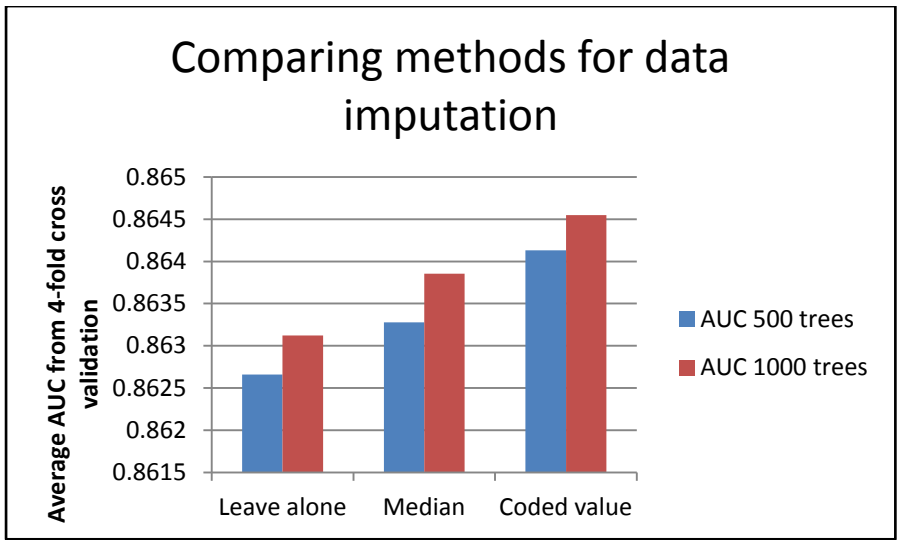

Figure 2: Comparing different ways of doing data imputation

3.2 Number of inputs sampled at each split

This input number refers to the number of inputs sampled at each iteration when building the forest. I tested the prediction performance with different sample size, all using the coded value heuristic for data imputation (since it gave the best performance among the three methods). From the results we see that a larger sampling size gives less accurate prediction. This could be because of overfitting to the dataset. The results of the tests can be seen in Figure 3.

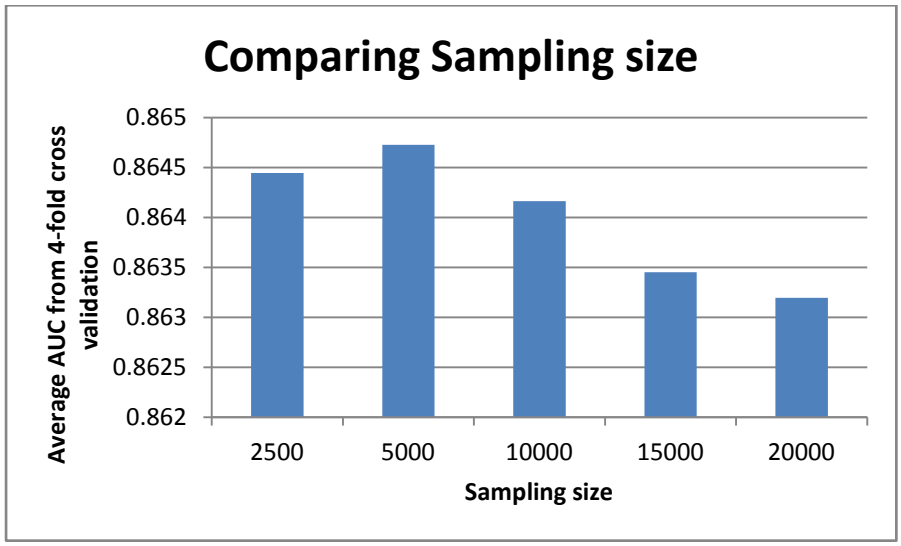

Figure 3: Comparing different sampling size

3.3 Number of trees grown

Lastly, from the best performers in the past two parameters, I ran a test for 500 and 1000 trees grown. Actually, most of the "learning" occurred within the first 350 trees that were grown, and only incremental changes appeared thereafter. Nonetheless, it appears that the performance was slightly better when we grew only 500 trees as opposed to 1000 trees. Again, I suspect this was due to overfitting of the data. Results are shown in Figure 4.

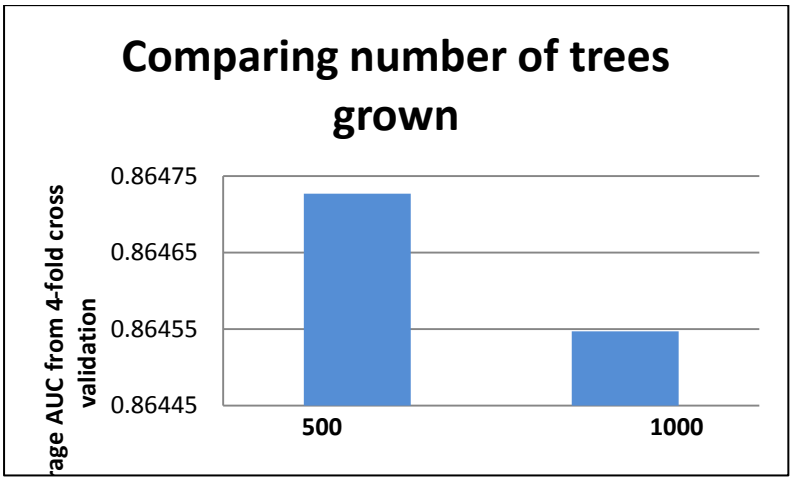

Figure 4: Comparing number of trees built in the forest

3.4 Submission to Kaggle

The above results were calculated from doing a 4-fold cross validation on the labeled data provided by Kaggle. However, after all the parameters were chosen and a "best" set was found, the parameters were used to train on the labeled training data, and used to predict the unlabelled test data. This was then submitted to Kaggle which did the final AUC scoring. Using the parameters chosen above, I got an AUC of 0.867262. As a reference, the top team got a much better result, an AUC score of 0.869558.

4 Discussion

Simply by tweaking a few parameters, I was able to get rather good prediction results on the dataset. This is an example of the strength of random forests: its default parameters are generally quite good. From discussions that occurred on Kaggle after the competition is over, some of the teams also using random forests were able to get much better results, with smarter ways of doing data imputation, and by combining random forests with other ensemble methods like gradient boosting.

5 References

1. Bastos, Joao. "Credit scoring with boosted decision trees." *Munich Personal RePEc Archive* 1 (2007). Print.
2. Chen, Chao, Andy Liaw, and Leo Breiman. "Using random forests to learn imbalanced data."
3. Dahinden, C. . "An improved Random Forests approach with application to the performance prediction challenge datasets." *Hands on Pattern Recognition* - (2009). Print.
4. "Description - Give Me Some Credit - Kaggle." *Data mining, forecasting and bioinformatics competitions on Kaggle*. N.p., n.d. Web. 17 Dec. 2011. <http://www.kaggle.com/c/GiveMeSomeCredit>.
5. Leo, Breiman. "Random Forests." *Machine Learning* 45.1 (2001): 5-32. Print.
6. "Random forest - Wikipedia, the free encyclopedia." *Wikipedia, the free encyclopedia*. N.p., n.d. Web. 17 Dec. 2011. <http://en.wikipedia.org/wiki/Random_forest>.
7. Robnik-Sikonja, M.. "Improving Random Forests." *Lecture Notes in Computer Science* 1.3201 (2004): 359-370. Print.