Data Mining

Assignment 1

Classification Trees, Bagging and Random Forests

report by

Miguel Â. Simões Valente, Riccardo Bassani and Samuel Meyer

Submitted to

Ad Feelders

on October 8, 2020,

first term

For further questions contact:
Miguel Â. Simões Valente: m.a.simoesvalente@students.uu.nl, 6876005
Riccardo Bassani: r.bassani@students.uu.nl, 6866840
Samuel Meyer: s.j.meyer2@students.uu.nl, 5648122

Miguel Â. Simões Valente
Riccardo Bassani
Samuel Meyer

Introduction

A classification tree is a predictive model that can be built from a dataset of observations, and that exploits the knowledge of attributes and labels of the observations in the dataset to assign labels to novel observations of which only the attributes are known. The model is structured like a tree, and each novel instance is sorted down the tree until it reaches a leaf node, which defines its label. The path, and therefore the reached leaf, is determined by split rules on the attributes specified by the not-terminal nodes [1].

Single classification trees are fairly simple and easily explainable, but are typically not top performers [2]. An improvement can be made by reducing the variance of the prediction [3].

This is achieved through Bagging [4], which consists of using a single training set to build several different trees and then considering all their predictions together. The construction of different trees is made possible by drawing M bootstrap samples from the training set. This means every sample contains as many observations as the training set, but the observations are drawn with replacement, so that a specific observation can be repeated or absent in a specific sample, and samples are generally different from each other. Different samples will lead to different trees, therefore to different predictions. The final prediction will be the average of the predictions in the regression case, or the majority vote in the classification case.

By creating more diversified trees, a smaller variance can be obtained, thus better performance. A third prediction method, Random Forest [5], exploits this idea. In order to "decorrelate" the trees, it extends Bagging by limiting the attributes that can be considered at each split to a random subset of the set of attributes. In particular, in our implementation the cardinality of this set was determined to be $6 \simeq \sqrt{41}$.

In this report we analyze the differences in performance of the tree approaches: Single Tree, Bagging, and Random Forest.

Data

In order to analyse the quality of the Single Tree, Bagging and Random Forest algorithms we used the package level data of the Eclipse Bug dataset [6].

This dataset contains two main groups of features, one derived from the structure of the software in the form of an abstract syntax tree and another one based on several complexity metrics. For our experiments, we ignored the features obtained from the abstract syntax tree and used solely the complexity metrics and the amount of pre-release defects. With the 41 numerical features selected we proceeded to train our classification algorithms in a binary classification task, to determine if bugs have been reported post-release of the software. Release 2.0 was used as the training set, and release 3.0 as the test set.

An assumption/transformation was made with regard to the data, in order to make it fit the binary classification problem: the exact number of post-release bugs is not taken into account.

Assumption 1-$\mathcal{A}$. *All post-release bug reports above 1 were changed to 1, so that a binary representation was obtained where 0 means no post-release bugs, and 1 means some post-release bug.*

Results

In order to compare the performance of Single Tree, Bagging and Random Forest, for each method we derived the confusion matrix (Table 1) and computed the accuracy, precision and recall on the test set (Table 2).

To assess whether the differences in accuracy shown in Table 2 are statistically significant, three McNemar's Tests were performed, comparing respectively Single Tree and Bagging, Single Tree and Random Forest, and Random Forest and Bagging. The p-values of the tests are reported in Table 3.

Table 1: Confusion Matrices

Pred. \ True	1	0
1	266	128
0	82	185

Single Tree

Pred. \ True	1	0
1	301	103
0	46	210

Bagging

Pred. \ True	1	0
1	283	93
0	65	220

Random Forest

Table 2: Performance Measures

Measure \ Method	**Single Tree**	**Bagging**	**Random Forest**
accuracy	0.6823	0.7746	0.7610
precision	0.5911	0.7694	0.7580
recall	0.6929	0.7830	0.7623

Table 3: Result comparison McNemar's Test

Methods	**Single Tree & Bagging**	**Single Tree & Random Forest**	**Bagging & Random Forest**
p-value	$2.2059 * 10^{-8}$	$3.8872 * 10^{-6}$	0.3681

The results of the McNemar's test show that the differences in accuracy between Bagging and Single Tree and between Random Forest and Single Tree are statistically significant ($\alpha = 0.05$). Therefore both Bagging and Random Forest can be considered an improvement over the Single Tree model. In the case of Bagging it can be observed as an increment of +0.092 (+13.5%) for the accuracy, +0.178 (+30.2%) for the precision, and +0.090 (+13.0%) for the recall. For Random Forest similar results are obtained, namely +0.079 (+11.5%) for the accuracy, +0.167 (+28.2%) for the precision, and +0.069 (+10.0%) for the recall. When comparing Bagging and Random Forest, instead, the difference in accuracy can not be considered as significant given the p-value of the corresponding McNemar's test. Thus, the two methods are considered to perform at the same level for this dataset and the given parameters.

To gain a further insight on the working mechanism of the Single Tree, we report the first two splits of the latter in Figure 1. Namely, this shows the split in the root node, and the split in its left child.

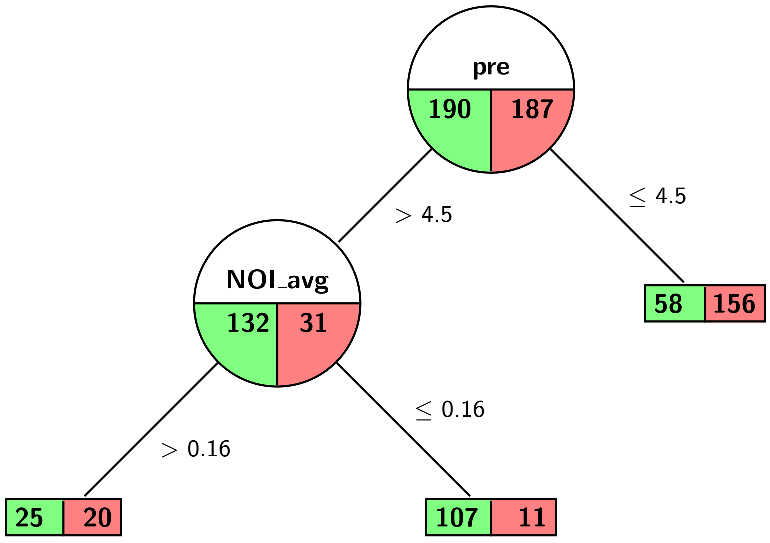

Figure 1: Shows the first two splits of the Single Tree. The green node sections show the number of instances for which there are bugs, prior to splitting in that node. The red node sections show the number of instances for which there are no bugs prior to splitting in that node. The white sections report the attributes on which the splits are performed: "pre" indicates the number of pre-release bugs, "NOI_avg" indicates the average number of interfaces.

Discussion

The analysis showed how using multiple trees instead of a single classification tree leads to better performance both in the case of Bagging and in the case of Random Forest. This is coherent with results found in the literature: the reduction in variance obtained through the use of multiple trees leads to a higher accuracy [3].

Despite being significantly worse than Bagging and Random Forest, the Single Tree accuracy is well above the chance level. To get a basic understanding of how the tree works, we can observe the tree obtained by pruning after the first two splits, as shown in Figure 1. The first split appears to be very reasonable: the split here is made on the attribute "pre", indicating the number of pre-release bugs. If this number is not higher than a threshold (4.5), then the right leaf is reached, and assigning the majority class results in predicting that there will not be post-release bugs. This is what can intuitively be expected, considering the low number of pre-release bugs.

In the case in which there are more than 4.5 pre-release bugs, a second non-terminal node is reached, where a split is made depending on the average number of interfaces. Here, regardless of which leaf is then reached, assigning the majority class results in predicting that there will be a post-release bug. This is exactly what is desired, since a high number of pre-release bugs will probably cause the presence of post-release bugs.

Deepening the analysis, it can be noticed that a low number of interfaces leads to a right leaf with much more cases of pre-release bugs than in the left leaf, which will result in more predictions of post-release bugs descending further down the tree. This also make sense, since a low number of interfaces is usually symptom of either lower complexity, or low quality code.

While both Bagging and Random Forest performed significantly better than Single Tree, the results of Bagging and Random Forest are quite similar to each other. However, earlier comparisons have indicated that the Random Forest method should perform better than Bagging [7].

This could be due to the presence of the attribute "pre", containing the value of the pre-release bugs. This attribute appears, not surprisingly, to be very important for predicting the presence of post-release bugs, as shown in the analysis of the Single Tree. Therefore, Bagging will most probably build trees where the first split is made on "pre". In the Random Forest case, on the other hand, the random selection of six attributes when choosing the best split could often exclude "pre" from the candidates. This would cause the performance of Random Forest to decrease, and to reach the level of Bagging.

In conclusion, the obtained Single Tree model appears to be a reasonable even though basic and not top performing classifier, which can be useful to identify the main causes of the presence of post-release bugs. Bagging and Random Forest, on the other hand, are more reliable classifiers, but the gain in accuracy comes at the cost of losing the ability to explain what led to the presence of bugs. The same conclusions are valid in general, with Single Trees being highly explainable but not extremely accurate, while multiple-trees model reaching higher accuracy values but sacrificing explainability.

References

[1] Tom M Mitchell. *Machine Learning, volume 1 of 1.* 1997.
[2] Wei-Yin Loh. "Fifty years of classification and regression trees". In: *International Statistical Review* 82.3 (2014), pp. 329–348.
[3] Thomas G Dietterich. "Ensemble methods in machine learning". In: *International workshop on multiple classifier systems.* Springer. 2000, pp. 1–15.
[4] Leo Breiman. "Bagging Predictors." In: *Mach. Learn.* 24.2 (1996), pp. 123–140.
[5] Leo Breiman. "Random Forests." In: *Mach. Learn.* 45.1 (2001), pp. 5–32.
[6] Thomas Zimmermann, Rahul Premraj, and Andreas Zeller. "Predicting defects for eclipse". In: *Third International Workshop on Predictor Models in Software Engineering (PROMISE'07: ICSE Workshops 2007).* IEEE. 2007, pp. 9–9.
[7] Robert E Banfield et al. "A comparison of decision tree ensemble creation techniques". In: *IEEE transactions on pattern analysis and machine intelligence* 29.1 (2006), pp. 173–180.