Utrecht University

Data Mining: Assignment 1

CLASSIFICATION TREES, BAGGING AND RANDOM FORESTS

Ana Borovac
6584446

Argyro (Iro) Sfoungari
6528015

October, 2018

Contents

1 Short Description of the Data **2**

2 Single Tree **3**

3 Single Tree, Bagging and Random Forests **4**

3.1 Single Tree 4

3.2 Bagging 4

3.3 Random Forests 5

4 Differences in Accuracy **6**

Abstract

In the first Data Mining assignment, we were asked to write two main functions tree.grow and tree.classify with specific input arguments each. Initially, we had to create a single tree, but then we had to expand it to Bagging and Random Forests by adding two auxiliary functions, tree.grow.bag and tree.classify.bag. The above code was created in order to analyse the Eclipse Bug Dataset. More specifically we had to do some tests in order to predict if any post release bugs have been reported. In other words, we created a single tree for both the training and the test set, while at the same time we followed the same procedure for Bagging and Random Forests. Finally, we examined accuracy, precision and recall for each dataset and case and we ended up to the following conclusion. With Z-test we proved that the difference in the classification accuracy between Single Tree and Random Forests is not statistically significant, in other cases it is.

1 Short Description of the Data

In the assignment we are asked to use the algorithms we have implemented to analyse the bug dataset of the Eclipse programming environment, before and after the release of each packet. According to the accompanying article [2], in order to explain the reason why some programs are more vulnerable to failure than others, we need to take a look at a bug database. Bug databases, list all the errors that occurred during the lifetime of a software. More specifically, the dataset contains information about the number of faults that have been reported in the first six months before and after release. As reported by the authors, metrics are useful for predicting the thickness of the defects. For the assignment we are going to use the 2.0 release as the training set and the 3.0 release as the test set and we are going to monitor if errors have been reported to later versions. We are asked to use the same metrics as the authors and the number of pre-release bugs.

These metrics are:

- FOUT (Number of method calls)
- MLOC (Method lines of code)
- NBD (Nested block depth)
- PAR (Number of parameters)
- VG (McCabe cyclomatic complexity)
- NOF (Number of fields)
- NOM (Number of methods)
- NSF (Number of static fields)
- NSM (Number of static methods)
- ACD (Number of anonymous type declarations)
- NOI (Number of interfaces)
- NOT (Number of classes)
- TLOC (Total lines of code)
- NOCU (Number of files)

And Pre-released defects: the defects that are observed during the development and testing of a program.

The training set consists of 377 samples:

- 187 samples labeled 0
- 190 samples labeled 1

The test set consists of 661 samples:

- 348 samples labeled 0
- 313 samples labeled 1

2 Single Tree

Below (figure 1) we have reproduced the first few splits of the single tree. We continue splitting until we get the representation of the left sub-tree. We have to process the given data, and draw to some conclusions in order to decide whether our classification rule makes sense. In our tree, the initial split is based on the number of pre-released defects that are detected during the development and testing of the program. As the split begins, we see that for the better case which is the one with the fewest pre-released defects (≤ 4.5), we have a shorter cyclomatic complexity (a software metric, used to indicate the complexity of a program) `VG_max`. This split seems quite reasonable if we think that with a more complex code we predict more bugs and vice versa, with fewer pre-released defects the complexity is reduced accordingly. Then, we split into the smaller total number of static methods `NSM_sum`, fewer total number of method calls `FOUT_sum`, smaller average of nested block depth `MBD_avg`, fewer maximum number of fields `NOF_max`, smaller average number of methods `NOM_avg` and static methods `NSM_avg` respectively and finally fewer lines of code. Consequently, we could support that the classification rule we created leads to reasonable results, as less complex code, gives less methods, method calls and other programming structures and finally fewer total lines of code.

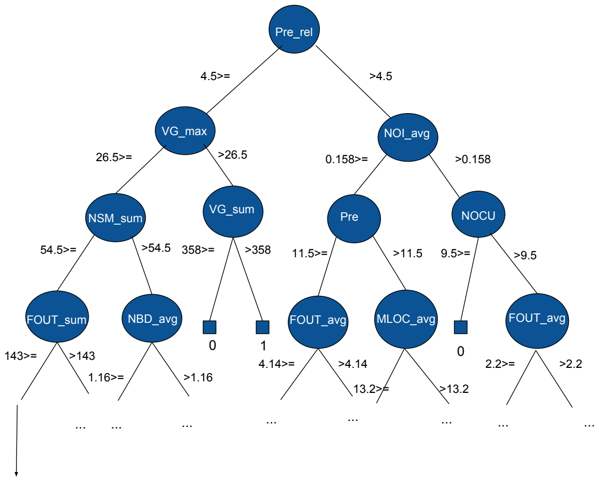

Figure 1: First splits of the trained single tree.

3 Single Tree, Bagging and Random Forests

3.1 Single Tree

We trained a single classification tree on the training set with `nmin = 15`, `minleaf = 5` and `nfeat = 41`. On the test set we got the following confusion matrix 1.

3.2 Bagging

We used bagging with `nmin = 15`, `minleaf = 5` and `nfeat = 41` and `m = 100`. On the test set we got the following confusion matrix 2.

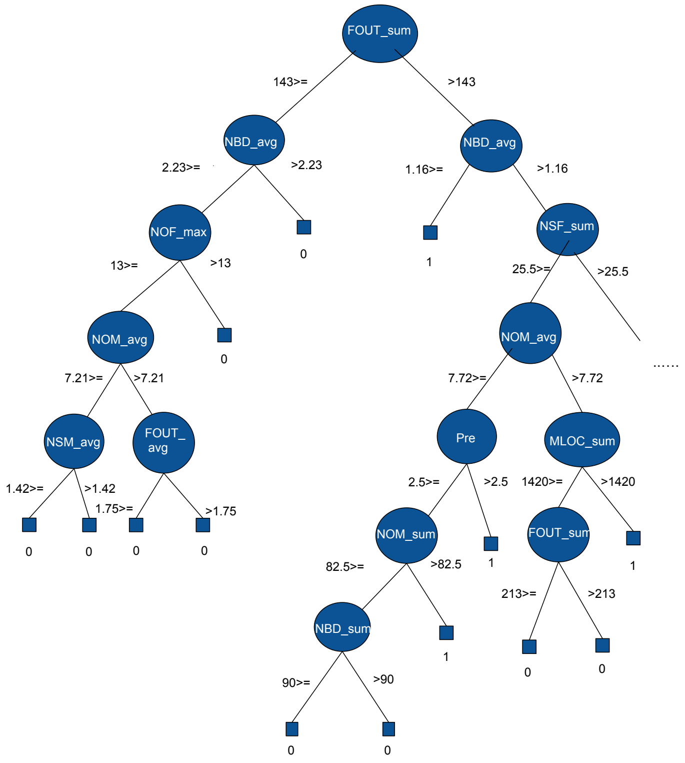

Figure 2: Splits

True class \ Predicted class	0	1
0	266	82
1	128	185

Table 1: Confusion matrix for a single tree run on the test set.

True class \ Predicted class	0	1
0	305	43
1	98	215

Table 2: Confusion matrix for a bagging run on the test set.

3.3 Random Forests

We used random forests with `nmin = 15`, `minleaf = 5` and `nfeat = 6`. On the test set we got the following confusion matrix 3.

True class \ Predicted class	0	1
0	268	80
1	120	193

Table 3: Confusion matrix for a random forests run on the test set.

4 Differences in Accuracy

First, let's have a look at table the 4. We can see that accuracy of Bagging is higher that accuracies of Single Tree and Random Forests. So, in the next step we would like to see if the difference is statistically significant.

	Accuracy	Precision	Recall
Single Tree	0,78	0,69	0,59
Bagging	0,91	0,83	0,69
Random Forests	0,79	0,71	0,62

Table 4: Accuracies, precisions and recalls of models: Single Tree, Bagging and Random Forests.

To test if the differences in accuracy found on the test set are statistically significant we used Z-test [1]:

$$Z = \frac{|f_{12} - f_{21}|}{\sqrt{f_{12} + f_{21}}}$$

where f_{12} denotes the number of samples that were classified correctly by the first algorithm and misclassified by the second algorithm (similar for f_{21}). If $Z \geq 1{,}96$ that means that a difference in the classification accuracy between the confusion matrices is statistically significant ($p \leq 0{,}05$).

In the table 5 there are Z-values for all the possible comparisons.

Compared algorithms	Z-value
Single Tree, Random Forests	0,81
Single Tree, Bagging	6,33
Bagging, Random Forests	5,45

Table 5: Calculated Z-values.

We can conclude that the difference in the classification accuracy between the confusion matrices of Single Tree and Random Forests is not statistically significant. In the other two cases, the difference is statistically significant.

References

[1] Pradeep Kumar, Rajendra Prasad, Arti Choudhary, Varun Narayan Mishra, Dileep Kumar Gupta, and Prashant K. Srivastava. A statisti-

cal significance of differences in classification accuracy of crop types using different classification algorithms. *Geocarto International*, 32(2):206–224, 2017.

[2] Thomas Zimmermann, Rahul Premraj, and Andreas Zeller. Predicting defects for eclipse [revised for dataset version 2.0a]. `www.st.cs.uni-saarland.de/softevo/bug-data/eclipse/promise2007-dataset-20a.pdf`. Online; accessed 10 October 2018.