Project 1

Dal Col Giada 0093122
Keller Dirk 4282264

14.10.2022

1 Description of the data

Our data consist in two data sets, one used as a training set and the other as test set. Both contain 40 variables, the metrics describing the components of Eclipse: some are related to methods, others to classes, files and packages and for all of them (except NOCU that counts the number of files in a package) we have an average, total and maximum value. For example, some features includes the "Number of method calls", "Number of static methods " or the "Total lines of code". In addition we used another binary variable which records if there is at least one prerelease bug. This concludes the set of features used to predict future errors in postreleased versions of the packages of Eclipse. The postreleased errors are included as the ground truth labels for the binary classification problem.

2 First splits

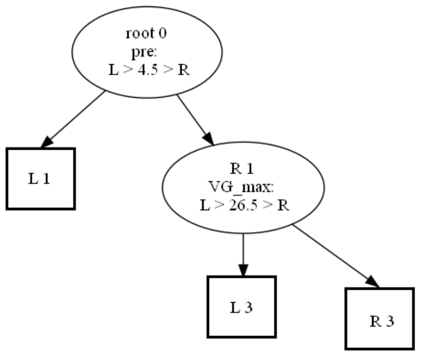

Figure 1: A picture of the first two splits of the single tree

In the simple tree (Figure **??**), the "pre" feature was considered to yield the best split of the root. Thus, the errors in the prereleased version seem to be

an important predictor of the errors occurring in the postreleased version. A package that is already demonstrated to be bug susceptible will be likely bug susceptible in later releases as well! Probably these packages constitute of an overall ill-designed architecture that might be inflexible. Therefore, a package with an error rate higher than 4.5 errors in the prereleased version is the predictor for identifying an error-prone postreleased version of the package.
The second order best feature of the right node is McCabe's cyclomatic complexity, which is a metric to indicate the complexity of a program. The cyclomatic complexity is demonstrated to be positively correlated with the frequency of defects occurring in a function or method. Thus, this metric is also used to identify risk of defects in a software program. Thus, the functions and methods that have the highest complexity tend to also contain the most defects. The tree considers the best split value of the cyclomatic complexity to be at 26.5, which is, according to McCabe, near to the lower boundary of a complex model (see Table 1). With respect to the tree construction, if a program has thus less than 4.5 errors in the prerelease and has a complexity score of less than 26.5, the tree will predict that the current instance - the software package - will be defect free in the postreleased version.

Simple procedure	More complex	Complex	Untestable
1 - 10	11 - 20	21 - 50	> 50
little risk	moderate risk	high risk	very high risk

Table 1: McCabe's interpretation of the cyclomatic score

3 Confusion matrices and quality measures

We have considered three different classifications with the same test set. First we used a single tree with some constraint on the leaves (minimum number of elements in a leaf and minimum number of elements to perform a split). Second, we classified with bagging using the same constraints and 100 bootstrap samples. Third, we used the random forest, where 6 variables are selected randomly for computing the best split. In Table 2 the confusion matrices obtained in the different cases.

Tree		Bagging		Random Forest	
250	98	302	46	285	63
102	211	99	214	93	220

Table 2: Confusion matrices

There is an improvement in the classification as the model increases in complexity: in fact single trees are not the best predictors, but averaging the result of multiple tree classifications reduces the error (using the majority vote);

suprisingly, random forest does not improve over bagging alone, since selecting randomly the features for the best split, reducing the correlation between trees, would be expected to improve the performance.
The quality measures for the three models are accuracy, precision and recall: the first relates the number of correct classifications (true positives and true negatives) to the total number of elements, the second relates the number of true positives to the number of elements predicted as defect-prone, while the last one relates the number of true positives to the number of elements that actually had defects. Ideally, we would like to have all this quantity as close as possible to one, which would result in a perfect classification. The obtained values are listed in Table 3.

	Accuracy	Precision	Recall
Tree	0.6974	0.6794	0.6837
Bagging	0.7761	0.8137	0.6837
Random Forest	0.7579	0.7666	0.7029

Table 3: Quality measures

There is a clear improvements in all three measures between the simple tree classification and the other two models, since we take the majority vote of all predictions of individual trees to get a model with lower variance.
Accuracy and precision instead decrease when using the random forest, because the size of the sample of the features is too small to have a high probability to have the best feature; it can happen that the random sampling of the features gives the ones that are ill suited for the split. Increasing "nfeat" to 10 substantially improves over 6 features with respect to all quality metrics, while still performing random forest sampling of features; hence the advantages of speed up in computation time and decorrelation of the individual trees is preserved.

4 Differences in accuracy

As seen before, we have some difference in the accuracy measure for the three models, with the bagging as the most accurate one, which means it's the predictor that makes less mistakes in classification of the test set.
In order to argue if these differences are statistically significant we used the McNemar's test: the null hypothesis states that the proportion of error in the test set is the same in the two classifications, which means basically that the two models disagree in the same way. If the null hypothesis can not be refused, then as a direct consequence we can argue that the difference in the accuracy is not statistically significant; in fact, having the same proportion of error means having the same proportion of correct prediction (accuracy).
Specifically, the test is based on the contingency table, a tabulation displaying the frequency of two binary variables counting the frequencies of the combina-

tions of correct/incorrect predictions in the two models.
In Table 4 the three contingency tables: the values actually used for computing the statistic for McNemar's test are the pairs T_i/F_j and F_i/T_j, that expresses the situation of one model classifying correctly and the other wrongly.

	T_2	F_2
T_1	492	27
F_1	94	111

(a) Tree vs. Bagging

	T_3	F_3
T_1	422	34
F_1	87	118

(b) Tree vs. Forest

	T_3	F_3
T_2	490	33
F_2	19	119

(c) Bagging vs. Forest

Table 4: Contingency tables

The test statistic is

$$\frac{(T_i/F_j - F_i/T_j)^2}{T_i/F_j + F_i/T_j}$$

with a Chi-Squared distribution with 1 degree of freedom. The p-values computed in the three tests are

Tree vs. Bagging	Tree vs. Random Forest	Bagging vs. Random Forest
$1.122713e^{-9}$	$14487233e^{-6}$	0.05220364

Clearly, with a significance level of 5% we can not refuse the null hypothesis in the third case: bagging and random forest have similar proportion of error on the test set and consequently they are similar in accuracy. On the contrary tree classification differs from the other two models for the proportion of error, since the small p-values suggest to refuse the null hypothesis, thus differences in accuracy are statistically significant.