Data Mining

Classification Trees, Bagging and Random Forests

Authors

Zwierd Grotenhuis	(6271669)
Dani Krebbers	(6215327)
Lianne Roest	(6200508)

October 9, 2020

Universiteit Utrecht

1 Data

A classification tree, a bagging algorithm and a random forests algorithm were trained on the Eclipse 2.0 dataset. Before the training could start, some preprocessing had to be performed. First, the predictor variables had to be extracted, which can be found in the article *"Predicting Defects for Eclipse"* by Zimmermann et al. as mentioned in the assignment. Second, the labelling of the data had to be defined, which can be found in the Eclipse 2.0 dataset under column "post". This labelling was not dichotomous, since the label described the amount of bugs. Therefore, it had to be transformed so that a sample has a label of 1 if it contained defects, and a 0 if not.

- Training data: (eclipse-metrics-packages-2.0) - 377 cases
- Testing data: (eclipse-metrics-packages-3.0) - 661 cases

2 Tree split

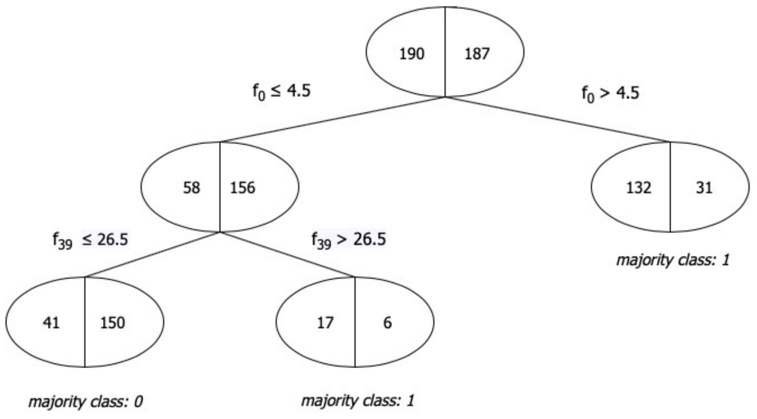

Figure 1: Simplified classification tree

In figure 1 we can see the simplified classification tree by applying the first two classifications rules. In each node the total number of occurrences is given for botch classes. With bugs on the left the class which is labeled as 1, without bugs on the right side with label 0.

The first classification rule is based on feature 0. From the training data can be derived that this feature matches the number of pre-release defects that were reported in the last six months before release of the package. This classification rule matches our expectation that the higher number of pre-release defects reported, the higher number of post-release defects will occur. This effect can also be inferred from the training data.

The second classification rule is based on feature 39, which corresponds with the maximum value of the McCabe cyclomatic complexity that can be found in the package. This measure looks at the complexity of a function. We expect that the maximum value is a better predictor than the average and total value. By using the total value of the McCabe cyclomatic complexity, the value will strongly correlate with the size of the package, which can be misleading. Using the average value of the McCabe cyclomatic complexity can give a distorted result, since methods with high complexities are less noticeable. This classification rule meets the expectation that a package with a high McCabe value will be more complex, thus resulting in more post-release bugs.

3 Results

3.1 Quality measures

	Single tree	Bagging	Random forest
Precision	0.693	0.84	0.784
Recall	0.591	0.652	0.696
Accuracy	0.682	0.776	0.766

Table 1: Quality measures for all three models

3.2 Confusion matrices

	Observed positive	Observed negative
Single Tree		
Predicted positive	266	82
Predicted negative	128	185
Bagging		
Predicted positive	309	39
Predicted negative	109	204
Random forest		
Predicted positive	288	60
Predicted negative	95	218

Table 2: Confusion matrices for all three models

4 Conclusion and discussion

Since we have 3 models with binary classification, we can create 3 contingency tables with which we can determine the difference in accuracy between the 3 models. With this table, we can use McNemar's test[1] to determine if the differ-

[1]`https://en.wikipedia.org/wiki/McNemar%27s_test`

ence between classification of two algorithms is significant. We will be looking at a significance level of 0.05, since this is standard for statistical analysis.

Below we have constructed the contingency tables for each combination of algorithms (tree, bagging, random forest).

	Bagging correct	Bagging wrong	Row total
Tree correct	423	90	513
Tree wrong	28	120	148
Column total	451	210	661

Table 3: Contingency table of Single Tree and Bagging

chi-squared: 31.53 & p-value: 1.96e-08

Since $p - value < 0.05$, we can conclude that the difference between the single tree algorithm and the random forest algorithm is significant.

	RF correct	RF wrong	Row total
Tree correct	415	91	506
Tree wrong	36	119	155
Column total	451	210	661

Table 4: Contingency table of Single Tree and Random Forest

chi-squared: 22.96 & p-value: 1.65e-06

Since $p - value < 0.05$, we can conclude that the difference between the single tree algorithm and the bagging algorithm is significant.

	RF correct	RF wrong	Row total
Bagging correct	473	33	506
Bagging wrong	40	115	155
Column total	513	148	661

Table 5: Contingency table of Bagging and Random Forest

chi-squared: 0.49 & p-value: 0.48

Since $p - value > 0.05$, we can conclude that the difference between the bagging algorithm and the random forest algorithm is not significant. This has some implications for future use of the algorithms. If the random forest and bagging algorithm indeed have the same performance, the random forest algorithm is preferred, since it takes up less memory and is faster to build. Based on our current research we can not confirm nor deny whether the algorithms have the same average performance.