ASSIGNMENT 1

Classification trees, bagging and random forests

UNIVERSITEIT UTRECHT

Noud Jan de Rijk – 5670721 Rose Mary Hulscher – 4272978 Yasmin van Dijk – 6014542

Data Mining

1. The data

The data set lists the number of pre- and post-release defects for every package and file in Eclipse releases in versions 2.0, 2.1, and 3.0. Historically, bug databases do not have information regarding how, where, and by whom bugs were solved. However, this data set contains metrics for the mapping of packages and classes to the number of defects that were reported in the first six months before and after release. The data was prepared by only selecting the appropriate predictors and thus removing features from the abstract syntax tree.

The removed features were features that counted syntactic elements of the package. The remaining features can be found in Table 1. These features are the '*pre*' feature (i.e. the amount of pre-release bugs) and the aggregated metric values for methods, classes, and files in a package. These aggregated values are condensed into average, maximum, and sum values for each predictor. Lastly, the data set featured the '*post*' attribute which contained the amount of post-release bugs. This value could be equal or greater than 0. However, the aim is to predict whether or not a package has bugs or not after it has been released, the '*post*' attribute was transformed with an *ifelse()* statement to contain either a zero (no bugs post-release) or a one (post-release bugs exist). This feature was used for testing the created prediction models.

pre	MLOC_avg	NOF_avg	NOM_max	NSF_sum	TLOC_avg
ACD_avg	MLOC_max	NOF_max	NOM_sum	NSM_avg	TLOC_max
ACD_max	MLOC_sum	NOF_sum	NOT_avg	NSM_max	TLOC_sum
ACD_sum	NBD_avg	NOI_avg	NOT_max	NSM_sum	VG_avg
FOUT_avg	NBD_max	NOI_max	NOT_sum	PAR_avg	VG_max
FOUT_max	NBD_sum	NOI_sum	NSF_avg	PAR_max	VG_sum
FOUT_sum	NOCU	NOM_avg	NSF_max	PAR_sum	

Table 1. Used predictors of the Eclipse data set

2. The split

In the single tree, the root node is split on the number of defects reported in the last six months before release, see figure 1. The majority of the cases with less than 4.5 pre-release defects have no post-release defects, the majority of the cases with more than 4.5 pre-release defects do have post-release defects. Assigning to the majority classes will give the following classification rules: If a case has less than 4.5 pre-release defects it will have no post-release defects, if a case has more than 4.5 pre-release defects it will have post-release defects. This makes sense, a higher number of pre-release defects means it likely there will be post-release defects.

The left child node of the root node is split on the maximal McCabe cyclomatic complexity. The majority of the cases with a McCabe cyclomatic complexity less than 26.5 have no post-release defects, the majority of the cases with a McCabe cyclomatic complexity higher than 26.5 do have post-release defects. Assigning to the majority classes will give the following classification rules: If a case has a McCabe cyclomatic complexity lower than 26.5 it will have no post-release defects, if a case has a McCabe cyclomatic complexity higher than 26.5 it will have post-release defects. This makes sense because a higher complexity means code is more error-prone, resulting in more defects.

The right child node of the root node is split on the average number of interfaces. The majority of the cases with an average number of interfaces lower than 0.158 have post-release defects, the majority of the cases with an average number of interfaces higher than 0.158 also have post-release defects. Assigning to the majority classes results in the following classification rules: If a case has an average number of interfaces lower than 0.158 it will have post-release defects. If a case has an average number of interfaces higher than 0.158 it will have post-release defects. This makes sense because it means the average number of interfaces doesn't correlate with the number of post-release defects. A high number of interfaces can mean the code is really complex, resulting in more defects. However, a high number of interfaces can also mean the code is well organized which would mean less defects.

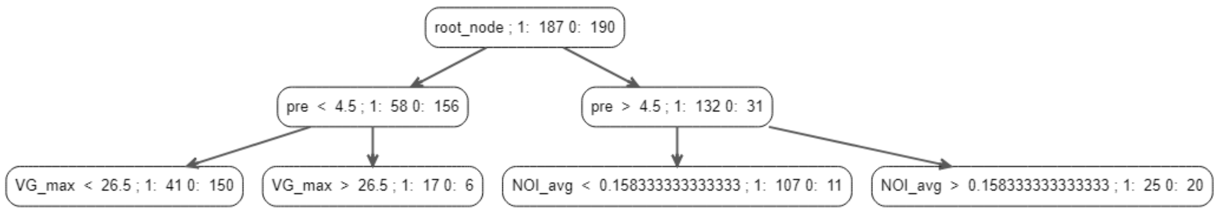
root_node ; 1: 187 0: 190
pre < 4.5 ; 1: 58 0: 156
pre > 4.5 ; 1: 132 0: 31
VG_max < 26.5 ; 1: 41 0: 150
VG_max > 26.5 ; 1: 17 0: 6
NOI_avg < 0.158333333333333 ; 1: 107 0: 11
NOI_avg > 0.158333333333333 ; 1: 25 0: 20

Figure 1: Overview of the splits

3. The Quality

3.1 Quality measures

To evaluate the performance of the prediction models, the following quality measures were used:

- Accuracy
- Precision
- Recall

Accuracy refers to the model's ability to correctly predict class labels for the complete testing data set. To compute this quality measure, the following formula was used:

$$\frac{TP + TN}{TP + TN + FP + FN}$$

Precision refers to the correctness of the model's predictions. That is, how many of the model's positive predictions actually are positive. To compute this quality measure, the following formula was used:

$$\frac{TP}{TP + FP}$$

Recall refers to the model's ability to predict true positives out of all positives of the testing data set. To compute this quality measure, the following formula was used:

$$\frac{TP}{TP + FN}$$

3.2 Confusion matrices and calculated quality measures for the prediction models

3.2.1 Confusion matrix of the single tree prediction model

269 *(TP)*	79 *(FP)*
126 *(FN)*	187 *(TN)*

3.2.1.1 Calculating the quality measures

Accuracy = $\frac{TP + TN}{TP + TN + FP + FN} \approx \frac{269 + 187}{269 + 187 + 79 + 126}$ = 0.690

Precision = $\frac{TP}{TP + FP}$= $\frac{269}{269 + 79}$≈ 0.773

Recall= $\frac{TP}{TP + FN}$= $\frac{269}{269 + 79}$≈ 0.681

3.2.2 Confusion matrix of the bagging prediction model

301 *(TP)*	47 *(FP)*
105 *(FN)*	208 *(TN)*

3.2.2.1 Calculating the quality measures

Accuracy = $\frac{TP + TN}{TP + TN + FP + FN}$≈ $\frac{301 + 208}{301 + 208 + 47 + 105}$≈ 0.770

Precision = $\frac{TP}{TP + FP}$= $\frac{301}{301 + 47}$≈ 0.865

Recall = $\frac{TP}{TP + FN}$= $\frac{301}{301 + 105}$≈ 0.741

3.2.3 Confusion matrix of the random Forest prediction model

292 *(TP)*	56 *(FP)*
93 *(FN)*	220 *(TN)*

3.2.3.1 Calculating the quality measures

Accuracy = $\frac{TP + TN}{TP + TN + FP + FN}$≈ $\frac{292 + 220}{292 + 220 + 56 + 93}$= 0.774

Precision = $\frac{TP}{TP + FP}$= $\frac{292}{292 + 56}$≈ 0.839

Recall = $\frac{TP}{TP + FN}$= $\frac{292}{292 + 56}$≈ 0.758

4. The test

4.1 Chosen statistical test

To evaluate the differences in accuracy between the predictions of the single tree, tree with bagging, and the random forest, McNemar's Test was chosen. This test compares attributes of a dichotomous nature (i.e. mutually exclusive) and checks if there is marginal homogeneity within the contingency table. In the case of our data set, this table would have an observed class and a predicted class with the labels: "Bugs" and "No bugs".

There are three main assumptions that the data must meet in order to effectively perform McNemar's test. These are:

1. There is a nominal variable with two categories and one independent variable connected to both categories.
2. Objects of one class can not appear in more than one group (i.e. they are mutually exclusive)
3. The sample must be a random sample.

These assumptions are met by our data set as the 'post' variable is a dichotomous variable as there are only two possible values: 0 and 1. Furthermore, the sample taken from the data set was taken at random.

4.2 Statistical significance

When performing McNemar's test, the statistical significance ($\boldsymbol{\alpha}$) was set at 0.10. This value was chosen because the test was performed in an undirected manner, meaning that the test was performed to see if the predictions of a certain model were equal or unequal to each other. Therefore, twice the default statistical significance was taken.

4.3 Hypotheses

For each test the following hypotheses were defined to determine whether or not the difference in accuracies of the two tests were statistically significant.

H_0: The difference between the accuracy of the first prediction model is equal to the accuracy of the second prediction model.
H_A: The difference between the accuracy of the first prediction model is different to the accuracy of the second prediction model.

4.4 Performing McNemar's Test

To perform McNemar's Test in R, the following function was used:

mcnemar.test(x, y, correct = TRUE)

The parameters are defined as follows:
x: The vector of predictions made by the first prediction model.
y: The vector of predictions made by the second prediction model.
correct = TRUE: This parameter states that continuity correction must be applied when computing the test statistic.

4.4.1 Single Tree & Bagging

The output of McNemar's test on the prediction vectors of the single tree prediction model and the bagging prediction model is:

McNemar's chi-squared = 0.82645, df = 1, p-value = 0.3633

Based on this result, it can be concluded that p (0.3633) > $\boldsymbol{\alpha}$ (0.10). Therefore, the difference between the accuracy of the single tree prediction model is not statistically significant to that of the bagging prediction model. This means that the null hypothesis should not be rejected.

4.4.2 Single Tree & Random Forest

The output of McNemar's test on the prediction vectors of the single tree prediction model and the random forest prediction model is:

McNemar's chi-squared = 0.66393, df = 1, p-value = 0.4152

Based on this result, it can be concluded that p (0.4152) > $\boldsymbol{\alpha}$ (0.10). Therefore, the difference between the accuracy of the single tree prediction model is not statistically significant to that of the random forest prediction model. This means that the null hypothesis should not be rejected.

4.4.3 Bagging & Random Forest

The output of McNemar's test on the prediction vectors of the bagging prediction model and the random forest prediction model is:

McNemar's chi-squared = 7.0175, df = 1, p-value = 0.008071

Based on this result, it can be concluded that p (0.008071) < $\boldsymbol{\alpha}$ (0.10). Therefore, the difference between the accuracy of the bagging prediction model is statistically significant to that of the random forest prediction model. This means that the null hypothesis should be rejected, and the alternative hypothesis could be adopted.

References

Picture on homepage: https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Fubuntupit.com%2Fwp-content%2Fuploads%2F2019%2F04%2Fsales-prediction.jpg&f=1&nofb=1