CS/SE 6356 Software Maintenance, Evolution & Re-engineering
Spring 2024
Assignment 6: Defect prediction

Deadline: **05/01/2024**

1. Goal of the assignment

In this assignment, you will build a defect predictor that automatically identifies defect-prone classes in Apache PDFBox. Each team must complete the following instructions. This is a team assignment. As before, you are NOT to discuss with other teams the results, but you can discuss technical issues.

2. Identifying defect-prone classes

You can use WEKA 3[1] to train and test your prediction model. Make sure you understand the input and output of WEKA. Use the WEKA self-tutorial and the defect prediction slides posted on eLearning. If you are more familiar with other ML library, feel free to use it, but explain how and what you did. It is important to note that while you may use different libraries to train and test your prediction model, you cannot use an existing tool to directly make the predictions for you.

You will collect data to train your model using the following versions **only**: 2.0.26, 2.0.27, 2.0.28 2.0.29, 2.0.30

Define each class in each version as a unique instance for training your model. For example, class C in versions 2.0.26 and 2.0.27 must be treated as different classes, hence you must define two separate classes (i.e., instances) in your training data (see the example below).

Determine whether each class in each version is buggy or not based on the number of bugs for the class between the current version and the version immediately before it. This will be the dependent variable for your prediction model (i.e., buggy = 1 or 0). For example, count the number of bugs for each class (from version 2.0.27) between versions 2.0.26 and 2.0.27, count the number of bugs for each class (from version 2.0.28) between versions 2.0.27 and 2.0.28, etc. If the number of bugs is greater than zero, then the class is buggy (buggy=1), otherwise the class is non-buggy (buggy=0). To compute the number of bugs in a class C, find all commits that change the file of class C between the two considered versions and link it to an issue in the issue tracker. If the issue is a bug report, then you mark/add one bug for class C.

For each software version, use the metrics tool you used in your previous assignments (or any other tool you may chose) to compute the following metrics on each class: WMC, LCOM, DCC, and DIT. These metrics will be the independent variables for your prediction model.

In the end, your training data should look like the following table:

Class-version	WMC	LCOM	DCC	DIT	Buggy?
org.apache.pdfbox.ClassA-2.0.26	0.5	0.2	2	3	0
org.apache.pdfbox.ClassB-2.0.26	2.5	1.2	3	5	1
org.apache.pdfbox.ClassA-2.0.27	2	1	1	4	1
…					

In this example, the metric values are made up and don't reflect the actual output range of the metrics. More importantly, note that class A is duplicated in versions 2.0.26 and 2.0.27, hence it is defined twice in the training data. Also, note that

[1] https://waikato.github.io/weka-site/index.html

class A is only buggy in version 2.0.27, which means that at least one bug report was found to impact the class because the report was associated to a commit that modified the file (between versions 2.0.26 and 2.0.27).

1.1 Training the classifier

I recommend using a simple classifier – J48 – which is a decision tree. If you use another ML library, choose a similar model. The steps below are for J48, but they are the same for any other decision tree classifier you may choose.

1. Train a J48 classifier implemented in WEKA using the default classifier parameters on the training data you collected. Use 10-fold cross-validation to train you model and report the training results reported by WEKA, including the J48 pruned tree, results summary, detailed accuracy by class, and confusion matrix.
2. Select the top **75** largest classes (based on LOC – lines of code) from the version 2.0.31 and report the results of the predictor for each of them in a table (i.e., predict if each class is buggy or not according to trained model). While you need to make the prediction for 75 classes only, you need to build the classifier based on all the classes.
3. Inspect the results. Establish, based on the results, which is the dominant variable for determining if a class is likely to be buggy or not. Explain your reasoning.
4. Computer the accuracy, recall, and precision for your predictor.

3. Resources

- The code repositories of PDFBox[2] is hosted on GitHub.
- Apache PDFBox[3] uses Jira as issue tracker.
- WEKA 3 is available online[4], including usage documentation.

4. Hints

- You can utilize the data/code from the previous assignments. To identify the number of bugs of each class, you need to associate classes with files, as much of the data from the previous assignments is at file level. Assume there is one class per file. If there are additional embedded classes in a file, ignore them. You can identify the main class in a file with a simple script matching the file name with the class name.
- When determining the number of bugs for a file/class in version N, count the number of bug reports fixed between version N-1 and N, which involved modifying the file/class (based on the commit log of the projects).
- Use simple text matching to link commits and bug reports, by finding the unique identifier of the reports in the commit descriptions, as you did in the previous assignment.
- As J48 (if that is your choice for a learning model) only works with nominal classes, you need to convert the dependent variable to a nominal variable (rather than numeric).
- You do not need to keep track of files/classes that are deleted and added again. Instead, treat every added class as a new one. Remember that classes across different version should be treated as separate classes or instances in your training data.
- If you wish, you can include additional metrics.

5. Deliverables

There are three deliverables for this assignment:

1. **Results report.** The report must be submitted as a PDF file on eLearning and should contain step descriptions, explanations, figures, and tables that support your answers. Explain the role and effort of each team member in the report.
2. **Code**. The source code you used to process the data and identify the defect-prone classes, in a zip file.
3. **Data**. The data that you collected and used to train your model.

[2] https://github.com/apache/pdfbox
[3] https://issues.apache.org/jira/projects/PDFBOX
[4] https://waikato.github.io/weka-site/index.html

6. Grading

Item	Points
Training data collected	30
Answers: buggy files, metrics, predictions.	50
Code to process the data and identify classes	20
Total	**100**

You will receive points off if:

- Your document contains typos and writing errors.
- Your explanations are unclear.