Replication of Defect Prediction Studies
Problems, Pitfalls and Recommendations

Thilo Mende
University of Bremen, Germany

PROMISE’10
12.09.2010
Replication is a waste of time.

"Replicability is not Reproducibility: Nor is it Good Science"

–Drummond (2009)
Replication is a waste of time.

"Replicability is not Reproducibility: Nor is it Good Science"
– Drummond (2009)

<table>
<thead>
<tr>
<th>Reproduction</th>
<th>Replication</th>
</tr>
</thead>
<tbody>
<tr>
<td>small changes</td>
<td>no changes</td>
</tr>
<tr>
<td>desirable</td>
<td>wasted effort</td>
</tr>
</tbody>
</table>
Replication is a waste of time.

"Replicability is not Reproducibility: Nor is it Good Science"

– Drummond (2009)

<table>
<thead>
<tr>
<th>Reproduction</th>
<th>Replication</th>
</tr>
</thead>
<tbody>
<tr>
<td>small changes</td>
<td>no changes</td>
</tr>
<tr>
<td>desirable</td>
<td>wasted effort</td>
</tr>
</tbody>
</table>

Replication → **identical** results
So why did I do this?
So why did I do this?

Prerequisite for:
- Reproducing results on other data sets
- Further analyses
Tosun et al.: "Validation of network measures as indicators of defective modules in software systems", PROMISE 2009.

Reproduction of Zimmermann and Nagappan (2008)
Data Sets

AR 3-5

Input: Complexity ✓
Input: Call-Graph ×
Dependent ✓

Eclipse

✓
×
?
## Experimental Setup

<table>
<thead>
<tr>
<th></th>
<th>Logistic Regression</th>
<th>Naive Bayes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformation</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>Algorithm</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Implementation</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Cutoff</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Evaluation</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Performance Measures</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Results

Logistic Regression

![Graph showing Recall, Precision, False Positive, and Balance for AR3, AR4, AR5, v2_0, and v2_1]
Results

Logistic Regression

Naive Bayes
Problems

- Class-Labels
- Cutoff
Problems

- Class-Labels
- Cutoff
- Partitions with no positives
- Large variation
- Class-Labels
- Cutoff
- Partitions with no positives
- Large variation
An Extensive Comparison of Bug Prediction Approaches

Marco D’Ambros, Michele Lanza
REVEAL & Faculty of Informatics
University of Lugano, Switzerland

Romano Robbes
Computer Science Department (DCC)
University of Chile, Santiago, Chile

Abstract—Reliably predicting software defects is one of software engineering’s holy grails. Researchers have devised and implemented a plethora of bug prediction approaches varying in terms of accuracy, complexity and the input data they require. However, the absence of an established benchmark makes it hard, if not impossible, to compare approaches. We present a benchmark for defect prediction, in the form of a publicly available data set consisting of several software systems, and provide an extensive comparison of the explanatory and predictive power of well-known bug prediction approaches, together with novel approaches we devised. Based on the results, we discuss the performance and stability of the approaches with respect to our benchmark and derive a number of insights on bug prediction models.

I. INTRODUCTION

Defect prediction has generated widespread interest for a considerable period of time. The driving scenario is resource allocation: Time and manpower being finite resources, it makes sense to assign personnel and/or resources to areas of a software system with a higher probable quantity of bugs. A variety of approaches have been proposed to tackle the problem, relying on diverse information, such as code metrics [1]-[8] (lines of code, complexity), process metrics [9]-[12] (number of changes, recent activity) or previous defects [13]-[18]. The jury is still out on the relative performance of these approaches. Most of them have been evaluated in isolation, or were compared to only few other approaches. Moreover, a significant portion of the evaluations cannot be reproduced since the data used by them came from commercial systems and is not available for public consumption. As a consequence, articles reached opposite conclusions. For example, in the case of size metrics, Gümeth et al. reported good results [6] unlike Fenton et al. [16]. What is missing is a baseline against which the approaches can be compared. We provide such a baseline by gathering an extensive dataset composed of several open-source systems. Our dataset contains the information required to evaluate several approaches across the bug prediction spectrum on a number of systems large enough to have confidence in the results. The contributions of this paper are:

• A public benchmark for defect prediction, containing enough data to evaluate several approaches. For five open-source software systems, we provide, over a five-year period, the following data: (1) process metrics on all the files of each system, (2) system metrics on bi-weekly versions of each system, (3) defect information related to each system file, and (4) bi-weekly models of each system version if new metrics need to be computed.

• The evaluation of a representative selection of defect prediction approaches from the literature.

• Two novel bug prediction approaches based on bi-weekly samples of the source code. The first measures code churn as deltas of source code metrics instead of line-based code churn. The second extends Hannon’s concept of entropy of changes [10] to source code metrics. These techniques provide the best and most stable prediction results in our comparison.

Structure of the paper: In Section II we present an overview of related work in defect prediction. We describe our benchmark and evaluation procedure in Section III. In Section IV, we detail the approaches that we reproduce and the ones that we introduce. We report on their performance in Section V. In Section VI, we discuss possible threats to the validity of our findings, and we conclude in Section VII.

II. DEFECT PREDICTION

We describe several approaches to defect prediction, the kind of data they require and the various data sets on which they were validated. All approaches require a defect archive to be validated, but do not necessarily require it to actually perform their analysis. When they do, we indicate it.

Change Log Approaches use information extracted from the versioning system, assuming that recently or frequently changed files are the most probable source of future bugs. Nagappan and Ball performed a study on the influence of code churn (i.e., the amount of change to the systems) on the defect density in Windows Server 2003. They found that relative code churn was a better predictor than absolute churn [9]. Hassan introduced the entropy of changes, a measure of the complexity of code changes [10]. Entropy was compared to amount of changes and the amount of previous bugs, and was found to be often better. The entropy metric was evaluated on six open-source systems: FreeBSD, NetBSD, OpenBSD, KDE, KOffice, and PostgreSQL. Moser et al. used metrics (including code churn, past bugs and refactoring, number of authors, file size and age, etc.), to predict the presence/absence of bugs in files of Eclipse [11].
## Setup

<table>
<thead>
<tr>
<th>Mapping to data sets</th>
<th>5 open-source systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Independent</td>
<td>25+6 sets</td>
</tr>
<tr>
<td>Dependent</td>
<td># of defects</td>
</tr>
<tr>
<td>Algorithm</td>
<td>generalized linear models</td>
</tr>
<tr>
<td>Implementation</td>
<td>R’s glm</td>
</tr>
<tr>
<td>Evaluation</td>
<td>50×90:10 split</td>
</tr>
<tr>
<td>Performance Measures</td>
<td>Adjusted $R^2$, Spearman’s $\rho$, custom scoring system</td>
</tr>
</tbody>
</table>
Results
Scoring system

- 24 equal
- 5 rounding errors
- 2 outliers
The Summary

Wasted Time?
Maybe, but necessary prerequisite.

Issues:
- Wutoffs
- Data transformations
- Some data sets too small → huge variance
- Implementation details
- Rounding
ties with spearman

bbb
The Summary

+ = ?

Issues:
- Cutoffs
- Data transformations
- Some data sets too small → huge variance
- Implementation details
  - rounding
  - ties with spearman
  - ...

Wasted Time?
Maybe' but necessary prerequisite. 

[Image of graph and papers]
The Summary

Issues:
- Cutoffs
- Data transformations
- Some data sets too small → huge variance
- Implementation details
  - rounding
  - ties with spearman
  - ...

Wasted Time?

Maybe' but necessary prerequisite...
The Summary

Issues:
- Cutoffs
- Data transformations
- Some data sets too small → huge variance
- Implementation details
  - rounding
  - ties with spearman
  - ...

Wasted Time? Maybe, but necessary prerequisite...
An Observation

Spearman correlation between # input variables and $R^2$:

$$\rho = 0.890$$
An Observation

Spearman correlation between \textit{# input variables} and $R^2$:

$$\rho = 0.890$$

Generate \textit{random} variables

LoC $\times$
Performance with Random Data

![Graph showing performance with random data for different datasets.](image-url)

- Equinox
- Eclipse
- Lucene
- Mylyn
- PDE

- $R^2$ vs. number of input variables

![Graph axes and labels](image-labels)
Compared to a Trivial Model

\[
\begin{array}{cccccc}
\text{eclipse} & \text{mylyn} & \text{equinox} & \text{pde} & \text{lucene} \\
\end{array}
\]

\[
\begin{array}{cccccc}
\delta \rho & 0.0 & 0.1 & 0.2 & 0.3 & 0.4 \\
\end{array}
\]
And now?

- Start sharing experiments?
- Or PROMISE repository on GitHub?
- ... publishing ‘official’ results
- To improve best practices, e.g.
  - Minimal set of necessary information
  - Random and trivial models
Replication of Defect Prediction Studies
Problems, Pitfalls and Recommendations

Thilo Mende, University of Bremen
tmende@informatik.uni-bremen.de