An Analysis of Software Defect Prediction Studies through Reproducibility and Replication

by

ZAHEED MAHMOOD

Submitted to the University of Hertfordshire in partial fulfilment
of the requirements of the degree of

DOCTOR OF PHILOSOPHY

School of Computer Sciences
University of Hertfordshire

March 08, 2018

Acknowledgements

First and foremost I thank God. I thank Him for bringing very special people around me to make my PhD a success. My beloved wife and son, you are everything to me and I thank you for putting up with all the very long hours while I was away. I will be very much around now and you cant get rid of me. To my dear family, if I were to mention just your names here, the page might not be enough, and if I were to mention all you have done for me, the entire dissertation is too small. My family my success, I thank you for always believing in me and supporting me.

David Bowes, you are more than a supervisor to me, without you, I would never have gotten this PhD. I have stumbled through this remarkable journey and in every circumstance, good and bad you had always pulled me up. I don't have the words to express my gratitude. I have learned so much from you, to put it simply you are the best. Peter Lane, you are this special friend to me. You have always believed in me and supported me all the time. You always had a special angle you see things and you shared and guided me with them. Bruce Christianson, if not for you I would not have been in UH at all. You have supported me immensely, personally, academically and have always given me your valuable time just to listen and point me in the right direction. The feedbacks and counsel you have given me are priceless. Tracy Hall, you have no idea how honoured I am to have worked with you. David, I have to thank you again for supporting me to work with Tracy. You have been very kind, very rigorous and every meeting we had I come out feeling, I just made an achievement by discussing with you. You have taught me how to be precise, given me the courage to make my points hard-hitting, it has been really an honour working with you. Yi Sun, you have always been encouraging and have supported me through a lot in all aspects of life. I truly appreciate your kindness and will never forget it. Neil Davey, talking to you had always given me happiness, it felt really good tapping into your experience academically and finding the best drink to have on our Thursday get-togethers.

To my guys from different continents Jean, Ankur, Alex whenever I remember you guys I just smile. I thank you for always being there for me, the help you have given me is invaluable, and you always have given me a listening ear. The fourth, Deepak I wish I could repay you for you not only been there for me but also my family, a special thank you to you and your family. You know there is no way I would finish this without dedicating a section to you Ronak, Iiulia, Weam, Maria, Parivash, Anuradha. I am really lucky to have shared unique friendship with you individually and collectively.

ABSTRACT

Context. Software defect prediction is essential in reducing software development costs and in helping companies save their reputation. Defect prediction uses mathematical models to identify patterns associated with defects within code. Resources spent reviewing the entire code can be minimised by focusing on defective parts of the code. Recent findings suggest many published prediction models may not be reliable. Critical scientific methods for identifying reliable research are Replication and Reproduction. Replication can test the external validity of studies while Reproduction can test their internal validity.

Aims. The aims of my dissertation are first to study the use and quality of replications and reproductions in defect prediction. Second, to identify factors that aid or hinder these scientific methods.

Methods. My methodology is based on tracking the replication of 208 defect prediction studies identified in a highly cited Systematic Literature Review (SLR) [Hall et al. 2012]. I analyse how often each of these 208 studies has been replicated and determine the type of replication carried out. I use quality, citation counts, publication venue, impact factor, and data availability from all the 208 papers to see if any of these factors are associated with the frequency with which they are replicated. I further reproduce the original studies that have been replicated in order to check their internal validity. Finally, I identify factors that affect reproducibility.

Results. Only 13 (6%) of the 208 studies are replicated, most of which fail a quality check. Of the 13 replicated original studies, 62% agree with their replications and 38% disagree. The main feature of a study associated with being replicated is that original papers appear in the Transactions of Software Engineering (TSE) journal. The number of citations an original paper had was also an indicator of the probability of being replicated. In addition, studies conducted using closed source data have more replications than those based on open source data. Of the 4 out of 5 papers I reproduced, their results differed with those of the original by more than 5%. Four factors are likely to have caused these failures: i) lack of a single version of the data initially used by the original; ii) the different dataset versions available have different properties that impact model performance; iii) unreported data preprocessing; and iv) inconsistent results from alternative versions of the same tools.

Conclusions. Very few defect prediction studies are replicated. The lack of replication and failure of reproduction means that it remains unclear how reliable defect prediction is. Further investigation into this failure provides key aspects researchers need to consider when designing primary studies, performing replication and reproduction studies. Finally, I provide practical steps for improving the likelihood of replication and the chances of validating a study by reporting key factors.

Contents

List of Figures

List of Tables

GLOSSARY

$quality_{4p}$ four phased quality check criteria.

Balance Is the percentage of defective instances (minority) in a dataset. When the proportion of defective instances makes up 50% of the dataset, balance equals 100%.

Benchmark Is a study derived from comparing and synthesising best practises from previous work intended for researchers to build upon.

Conclusion Stability Is when a claim in one situation can hold in multiple situations. In this work, the term is discovered to be an equivalent of replicable.

Control Is an act of changing a specific component of an experiment while keeping other components constant in order to observe the effect that the changed component has on results.

Imbalance Is the proportion of defective instances in a dataset. It indicates a dataset has balance below 100%.

Original An original study also called primary study.

Replicate Is to repeat an existing study by making various changes and observing whether the same claims hold.

Replicator A study that replicates an original study.

Reproduce Is to repeat an existing study as close as possible and observe whether the same results are achieved.

Researcher bias Is the selective observation and recording of information, where researchers consciously or unconsciously allow their personal views to affect data interpretation and research approaches.

Reverse bias It is when researchers perform a biased replication of an original study, and consequently, draw conclusions that invalidate a valid original study.

Sources of variabilty factors affecting prediction model performance.

1. Introduction

1.1 Thesis Statement

In this dissertation inherent threats to software defect prediction research are uncovered and described. Scientifically, it would be beneficial for practitioners to reproduce and replicate all important results. So far, there have been limited numbers of such studies, especially on high quality original papers. In my work, I have looked at all existing studies of this nature and analysed them: I have found examples of good practice in their methodologies, but also examples of bad practice. I have also attempted independent reproduction of five studies. Based on my analysis and practical work, I have uncovered some issues with existing studies and attempts. One outcome of this work is a set of practical steps to help support and encourage more and better quality reproductions and replications in the future.

1.2 Background

When software developers write code for implementing software, mistakes can be made. Mistakes are often errors and manifestation of those errors lead to defective software [IEE 2010]. These defects may be detected through code inspection which requires intensive manual effort and unit testing [Runeson and Andrews 2003, Hovemeyer and Pugh 2004]. Defects missed by developers may cause the software to fail at the hands of the customer. Therefore, significant costs may be incurred when manual inspections do not find these defects. When software fails it becomes even more costly to companies and customers. These costs reach up to $50billion per year, in the US [Levinson 2001, Tassey 2008]. Consequently, substantial investments (research grants and research efforts) have been channelled towards automated techniques. Automating the process of finding defects has the potential to save inspection time and effort

for developers and companies while improving software quality [Hovemeyer and Pugh 2004]. So, the idea is to predict whether a new piece of software might be defective based on what happened in the past. Typically defect prediction can be viewed as a set of processes. Data is collected from source code and fed into a prediction model. The model then predicts whether there are defects in that piece of code that the model has not seen before. The performance of the prediction model is then measured to see how well the model was able to find the code that was defective or not. All studies are only looking at the construction and evaluation process on known data.

The performance of a defect prediction model is important. A developer would prefer a model that accurately finds defective code so that review time and effort are focused on those parts. Shepperd et al. [2014] meta-analysed 600 prediction models from a set of original studies taken from [Hall et al. 2012] systematic literature review. The study analysed what impact four important factors of a prediction model could have on its performance. The four factors and their impact on model performance are:

- the research group that performed the study (up to 31.01%)
- the dataset used (up to 31%)
- the metrics, i.e., input to the algorithm (up to 12.44%)
- the algorithm (up to 8.23%).

Research group had the most impact (up to 31%) on prediction model performance. Based on this finding Madeyski and Kitchenham [2017] remark that one would assume that the main topic in defect prediction (the algorithm) would have the most impact on model performance but it only accounts for 1.3%. So, *"it matters more who does the work than what was done"* [Shepperd et al. 2014]. Perhaps the most critical problem suggested by Shepperd et al. [2014]'s results is that a study done by a research group may not be repeatable by another group: This questions the value of defect prediction models to companies and their customers. If practitioners are meant to use such models to predict defects that could

lead to software failure [Fenton and Neil 1999] - at the very least these models should be repeatable[1] by any interested party.

In other scientific fields replication hashas helped confirm many important findings. These findings changed the very fundamental theories of particular phenomena. Theories that for many years have provided direction in those fields and were found to be false through replication. For example, the fundamental theory in Physics that more precisely, no information travels faster than particles of light has been questioned. Marangos [2000] explains that textbooks may be wrong about this speed of light. In fact, Mugnai et al. [2000] after several replications of that experiment showed that there are particles that travel 7 times faster than the particles of light. Another example is the cold fusion experiment. Fleischmann et al. [1989] experiments were to find excess heat, nuclear byproducts, from the surface of normal water. This could have disrupted the energy sector providing very cheap and sufficient energy. McKubre [2008] reported that Lonchampt et al. [1996] attempted to master the experiments through replicating the experiments as close as possible. Lonchampt et al. [1996] discovered important factors such as the calibrating of the temperature must be at a certain range for the experiment to be successful. Primary studies make discoveries providing replication studies opportunities to support or invalidate the primary results.

To find replication literature surrounding defect prediction, I used a search string (software AND defect OR fault OR error OR bug AND prediction OR detection AND replication OR reproduce). In the literature, multiple words are used to refer to the same entity (error, bug, defect, fault). I ran the search multiple times, each time adding the defect terms. So far, my search returned few papers outside the replication papers I later found in chapter 4. That is the papers that replicated the original studies identified by Hall et al. [2012]. It is noteworthy that no date restriction applies to my search. Most of the

[1] Depending on the context the word 'repeat' is used, the term can be 'reproduce' or 'replicate'. [Leek and Peng 2015, Madeyski and Kitchenham 2017] define reproduce as repeating an experiment using the same data, tools and method to get the exact results (i.e. can be verified which has to do with credibility and trustworthiness). While replicate means to repeat a study using different data, tools and method (i.e. checks for consistency not getting the exact result which has to do with generalisability and stability of results in multiple environments).

recent literature on replication and reproduction in defect prediction is in our paper (section A.1). Now, I discuss the articles I found that are not within the scope of the systematic review in chapter 4.

Kamsties and Lott [1995] is one of the earliest replications in defect prediction. The study replicated Basili and Selby [1987] and then made additions. The original looked at three ways to find defects using (32 professionals and 42 advanced students). The replication used 50 (C programming) students as subjects. The students gathered data about faults and failures (in three ways functional testing, structural testing, and code reading). The findings were that all three methods found defects, but the functional testing was more efficient for students (an agreement with the original) to find defects. Years later the same original study Basili and Selby [1987] was replicated by Wood et al. [1997]. The replication found out combining the three techniques is a more effective defect detection achieved consistently; this finding individually agrees with the three detection methods and also discovered a new result. Porter and Votta [1998] performed an initial study of how subjects (48 graduate students) can find defects. In the same study Porter and Votta [1998] carried out an internal replication. Internal replication is a replication done by the same authors but different context. The internal replication used 18 professional developers from Lucent Technologies. The findings are that it is not always necessary to use professional developers for research (they are expensive) since the graduates' fault detection rate was at par statistically with those of the professionals. Lanubile and Visaggio replicated Porter et al. [1995] on 3 defect detection approaches (Ad Hoc, Checklist, and Defect-based Scenario) for software requirements inspections. The Ad hoc approach is when reviewers use their experience to identify defects without any guidance. The checklist is when reviewers follow a list based on previous inspections to provide answers. The scenario-based approach creates a model of a particular class of defect. Questions are created based on that particular model and answered by a reviewer. All reviewers then combine these multiple and different scenarios. The replication reported a partial agreement. The replication results disagreed with defect-based scenario performing better than others as reported in the original. However, the replication agreed

that collection meetings do not improve defect detection rate. The original also provided an experimental kit. The replication used different subjects than the original. The original also performed two internal replications (same authors but used two different subjects to do the detection). Sandahl et al. [1998] also replicated Porter et al. [1995]. In the replication, they used less experienced subjects. The findings did not support that scenario-based detection method is superior to other methods. The result of Sandahl et al. [1998] is a disagreement with the original Porter et al. [1995] but an agreement with the first replication [Lanubile and Visaggio].

A trend in these studies shows that it is crucial to replicate studies. Replication can be by the same authors or not and with the same or different population to improve the credibility of results and make discoveries. In particular multiple replications of a single original study continuously corroborates, disagrees and produces new findings. The importance of replication suggests the need for matching the number of original studies with continuous replications to provide more reliable results for the foundations of scientific fields.

Therefore, in all fields of science having studies that replicate other studies is important [Ioannidis 2017]: Because *"A culture that values and practices reproducible science can push out the boundaries of knowledge with confidence that new discoveries have potential to lead to new knowledge"* otherwise *"an absence of replication and verification will lead to a published literature that misrepresents reality"* [Errington et al. 2014]. Undoubtedly, replication is at the cornerstone of scientific progress.

As a result of the importance of replication in science the overall aim of this dissertation is to gauge how repeatable defect prediction studies are. To our knowledge there is no study that systematically looked at reproducibility and replicability of defect prediction studies. Therefore, in this dissertation so far, I identify the following research questions and provide the rationale behind them. The questions are based on the claim by Shepperd et al. [2014] that research may not be replicable and the importance of

replication practices in other scientific disciplines. It is noteworthy that Shepperd et al. [2014] analysed studies from the 208 studies identified by Hall et al. [2012]. The Hall et al. [2012] SLR provides the highest number of papers analysed in Software Engineering SLRs that I found. All the SLRs have the following number of studies: [Catal and Diri 2009] 74 papers, [Hall et al. 2012] 208 papers, [Radjenović et al. 2013] 106 papers, [Malhotra 2015] 64 papers, [Wahono 2015] 71 papers, and [Hosseini et al. 2017] 30 papers respectively. As such, I scope my analysis and reproduction approach to the most representative set of studies, 208 papers in defect prediction taken from the Hall et al. [2012] SLR. Here are my research questions:

RQ1: What proportion of defect prediction studies is replicated?

I reported replication examples in other scientific fields (and the flaws discovered) in this section. Based on these examples it is important to know if original studies are replicated in defect prediction to confirm findings or find mistakes and make new discoveries.

RQ2: What types of replications are performed?

At least if these replications occur it is important to identify how replications are performed. My approach will then provide best practices for researchers to adopt when performing replications.

RQ3: What features of a defect prediction study make it likely to be replicated?

It is important to identify the characteristics of original studies which are subsequently associated with replication. When these characteristics are adopted, researchers could produce more repeatable and verifiable research.

RQ4: Do original and replication studies in defect prediction agree?

Finding replications and defining how they are done is not enough. Determining the consistency of the findings - agreement between an original and its replication - is key to understanding the reliability of a field.

RQ5: What factors are likely to affect reproducibility in defect prediction studies?

Based on the context of Software Engineering - where tools and data are accessible - to what extent do original studies produce internally reliable experiments?

1.3 Contributions to Knowledge

My contributions in this dissertation include the identification of studies that have been replicated and how reliable their findings are. On the other hand it highlights a large number of studies that have not been replicated, an indication of uncertainty in the reliability of defect prediction research. The chapters (4, 5, and 6) provide influential factors on model performance which affect the ability of original studies to be reproduced and replicated. Finally, chapter 7 contributes practical recommendations to help support and encourage more and better quality reproductions and replications.

1.4 Structure of this Dissertation

The structure of this dissertation is as follows:

- chapter 2 gives a background to software defect prediction. The chapter defines replication and reproduction and examines how they are used in different disciplines.

- chapter 3 describes the methodology developed in this dissertation. The methodology is in two parts 1) a systematic study of original studies that have been replicated and their level of agreement. 2) a reproduction of papers that have been replicated and how close I get to agreeing with them. Unmatched results are then further investigated to find explanations for the differences in results.

- chapter 4 applies the first part of the methodology. The chapter identifies the number of studies replicated and their level of agreement. The chapter also identifies factors associated with replicated studies that lead to a paper being replicated. The results in this chapter answer RQs (1-4).

- chapter 5 applies the second part of my methodology. The chapter describes my reproduction of the 5 original studies that have been replicated. It also identifies potential factors causing differences in the results. The findings in this chapter contribute towards answering RQ5.
- chapter 6 describes further experiments performed to investigate the potential factors that may have lead to the reproducibility failures. The results in this chapter adds to the results in the previous chapter to answer RQ5.
- chapter 7 summarises all the work done and discusses the implications of my findings in light of research practices. The chapter also provides a list of practical recommendations to improve the state of research, reproduction and replication. Finally, the chapter suggests important future work based on my contributions.

2. Literature Review

The purpose of this chapter is to provide a literature review of the following concepts: defect prediction, studies on the unreliability of defect prediction, and the scientific principles of replicability and reproducibility. Prediction is about being able to suggest what might happen in the future based on what you know in the past. Defect prediction is a process by which data is collected from source code and input into a model. The model then predicts whether there are defects in the code. A simplistic overview of defect prediction flow is given by Kim et al. [2011], see Figure 2.1.

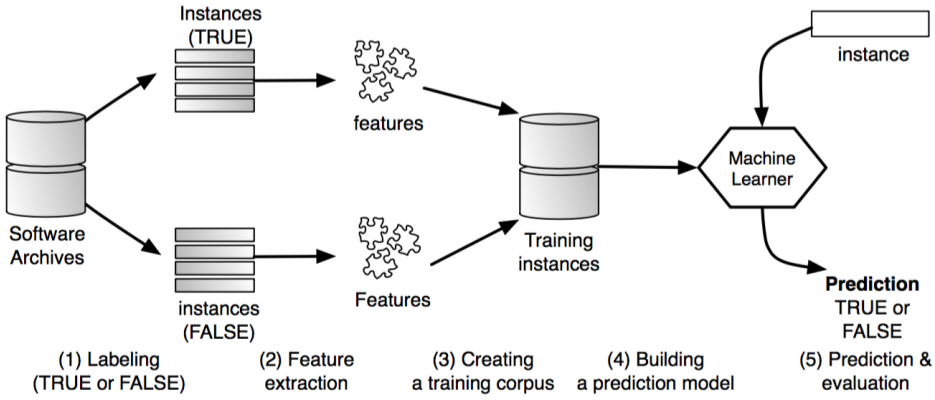


Figure 2.1: Source code are kept in software archives. Defective and non- defective code are extracted and labelled accordingly. Software is measured in different ways to represent features. The features are calculated from the defective and non defective code to make training instances. The model is trained on that data and used on unknown data to make predictions.[Kim et al. 2011].

I am now going detail how to collect data, build models and evaluate the results. Perhaps suggesting along the way how defect prediction may not be reliable and therefore requires reproduction and replication.

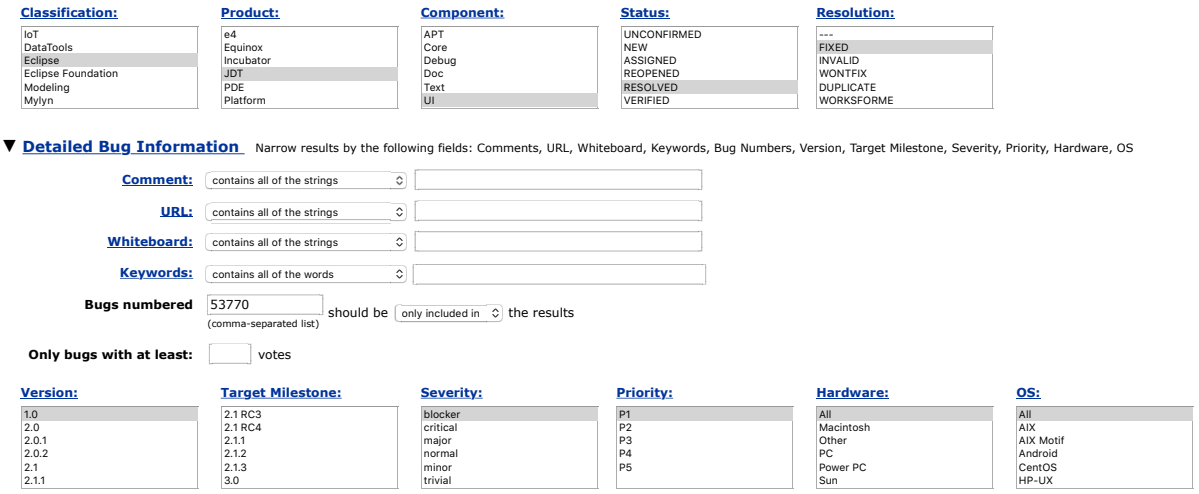
2.1 Collecting data for defect prediction

This section will describe how data is collected from the source code repository. After collecting the data it is organised in a suitable format for building the models. The organised data is then stored in a data sharing archive, also called defect data repositories.

Creation Stage:

First of all a set of requirements for a piece of software is provided to the developers. The developers write code to build the software. Errors made by developers when writing the code may be found during software testing. An unexpected behaviour of the system in operation means there is a defect. That defect is then reported to the Bug Tracking System (BTS). The BTS is a database that keeps records of reported bugs that occurred while the software was in operation (after the software was released). Each bug has a bug ID. Figure 2.2a below shows a bug report with bug ID 5377. The report also shows the date the bug was reported, the type of product worked on (eclipse JDT) and the developer it was assigned to (i.e. Martin) to fix. Figure 2.2b shows the BTS search parameters used for finding and extracting bug information to a suitable format for analysis (a XML or CSV are popular formats). Once the bug is 'Fixed' the status is changed to 'Resolved'. The code used by the developer to fix the bug is available in the Version Control System (VCS). A VCS tracks code changes, some of which fix the bug. Changes are 'committed' to the VCS. The automatic linking from bug database to version control is pioneered by [Śliwerski, Zimmermann, and Zeller 2005], the SZZ algorithm. Commit messages are provided at the time of the commit. The VCS keeps the developer's name, fixed code, bug ID of fixed bug [Davies et al. 2014]. For example in Figure 2.3a the commit message says 'Fix for 53770' making that piece linkable to the bug 53770 in the BTS (Figure 2.2a). The change made to that (.java file) is shown in Figure 2.3b. In revision 1.31 of Figure 2.3b there was no code and the fix are the 9 added lines in revision 1.32 which fixed the bug. When a bug ID is omitted in the commit, the loss of that type of bug (e.g. memory related

(a) Bugzilla bug report

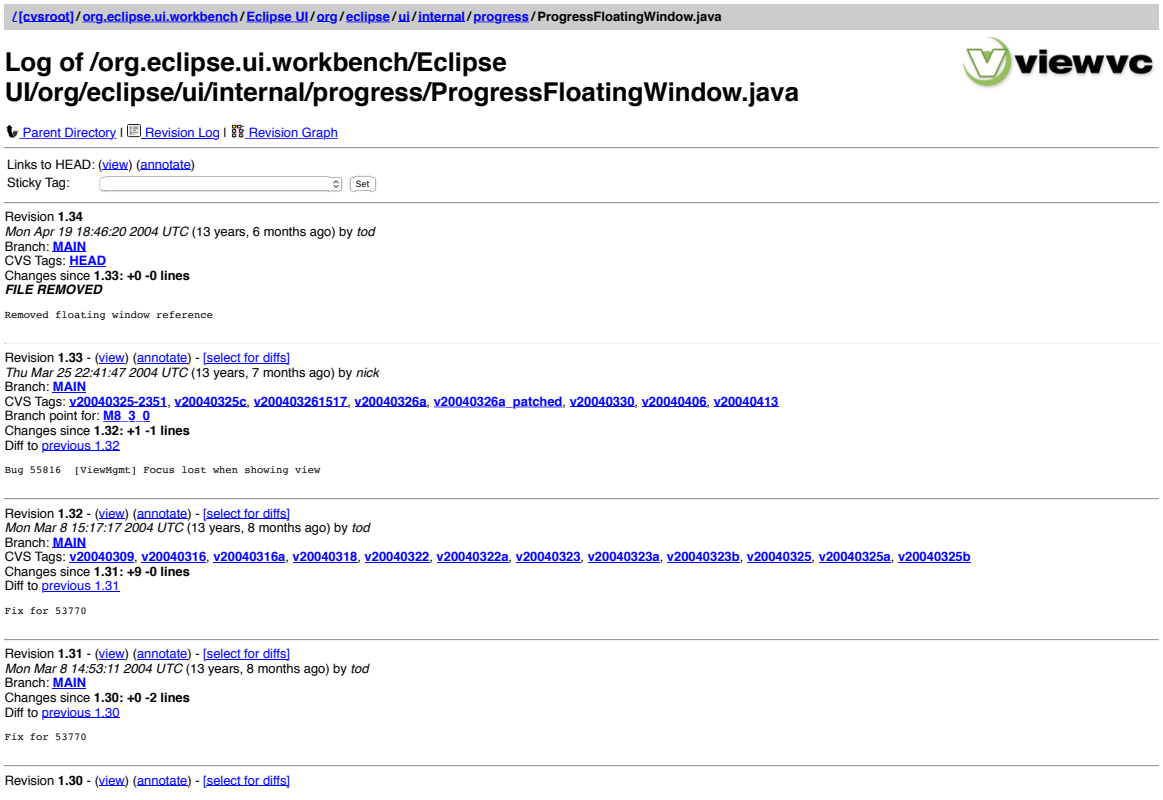


(b) Bugzilla search interface

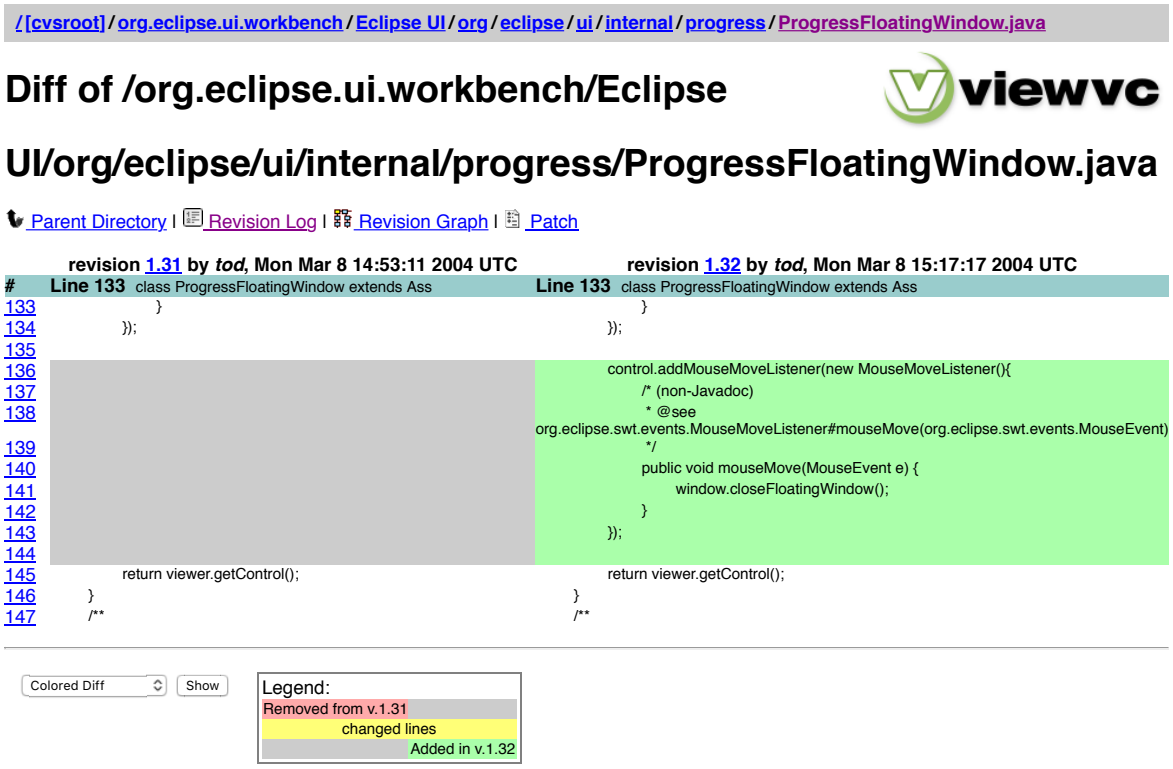
Figure 2.2: An example of a bug report (a) from Bugzilla database and its search interface (b). It is used for tracking defects that manifest when a software is running.

bug) occurs [Davies et al. 2014]; this human error begins to threaten the validity of the defect data, based on the certain type of bugs being missed. The defect data may not have sufficient representation of the bugs in real life. The under representation of some bugs in the dataset may also not allow generalisable models to be built. Reproducing the datasets may become more difficult when this omitted information is not reported accurately.

Linkage Stage: The linkage technique reported by [Śliwerski, Zimmermann, and Zeller 2005] (SZZ) uses a pattern matching algorithm (regular expressions), recently improved by Shippey et al. [2016], to match bug ID existent in both BTS (Figure 2.2a) report and a VCS commit message (Figure 2.3a) saved in commit logs. The successful linkage shows the code change that fixed the bug. Running a VCS 'diff'



(a) Commits written after fixing bug 53770



(b) Difference between old and new versions of the same .java file

Figure 2.3: ViewVC report-like tool is used to navigate through the source code repository

command gives previous changes of the code which is then used to find where the defective code was first introduced. According to Bird et al. [2009] defect linking is inconsistent due to omission of bug ID's by developers (creation stage Table 2.6) when committing a bug fix. Defect data built could over or under-represent certain types of defects. However the linkage inconsistency can be addressed manually as pointed out in [Bird et al. 2009]. Davies et al. [2014] manually simulated two linkage approaches and concluded that combining both gives better linkage.

Metrics Stage: Measurement is an important aspect to development. Roberts' book [Roberts 1979] on the theory of measurement provides the concepts of measurements in science and social sciences. How things are measured plays an important role in determining the development of a field. In the book, he further elaborates on the concepts. Examples of measurements concepts include: the physical, such as temperature; preference, such as loudness; and societal problems, such as noise pollution. The important idea is that taking measurements produces results and the results help in decision making. For instance, due to extremely low temperatures drivers could be urged to drive only when there is an emergency.

Baker et al. [1990] report that measurement theory [Roberts 1979] provides a way of categorising intuitive and meaningful characteristics of objects and events in software engineering. For example length of source code. The length can be represented using formal models such as integer. Length can then be ordered and also used to compare two programs in terms of their relative length. If a unit has the ability to be represented like 'length' taken from an aspect of software then it can be called a software measure, otherwise a metric if the mapping is arbitrary [Baker et al. 1990]. Software metric can be defined as *"a measure derived from the requirements, specification, design, code, or documentation of a computer program"*[Curtis 1983]. When software grows in size and usage it can be complex. Software complexity is an aspect of software that can be measured. There are different ways to measure software complexity. For example, [McCabe 1976] cyclomatic complexity measures. Fenton and Neil [1999] reported that the earliest use of size and complexity metrics was by [Akiyama 1971] for predicting the number of

```
1  /**
2    * Writes out 2D array row by row of the results for all classifiers on all datasets
3  */
4  public void writeResults() throws FileNotFoundException
5  {
6         PrintStream printResults = new PrintStream(new File(path));
7
8         printResults.print("classifiers ");
9
10        for (String dn : datasetNames)
11        {
12            printResults.print(", " + dn);
13        }
14            printResults.println();
15
16        for (int i = 0; i < results.length; i++)
17        {
18            printResults.print(classifierNames[i]);
19
20            for (int p = 0; p < datasetNames.length; p++)
21            {
22                printResults.print(" , " + results[i][p]);
23            }
24                printResults.println();
25        }
26                printResults.close();
27 }
```

Figure 2.4: An example Java code used to show how McCabe complexity metric is calculated. The code prints out the results of a defect prediction experiment of multiple classifiers that make predictions on multiple datasets

defects in a Fujitsu system. The most popular method-level metrics are McCabe and Halstead metrics [McCabe 1976, Halstead 1977]. I give a small example of how to calculate McCabes cyclomatic complexity metric in Figure 2.4. The Figure 2.4 shows a `writeResults()` method which is part of its class called `PrintWriter` which I wrote to print out the results of my experiments in chapter 5. From lines 6-26 the code is executed sequentially from top to bottom. The `for` statement is a statement that controls that flow followed by a condition to execute the block of code. McCabe's cyclomatic complexity counts the number of control paths in the `writeResults()` method. There are 3 control statements (`for`) and the starting point to the method (1), in total McCabe cyclomatic complexity denoted by $V(g) = 4$. The total lines of code is 20 LOC (executable lines between 4 and 27). These metrics can be calculated from the linked code (from the linkage stage) which contains defective and non-defective code. The calculation gives output of the metrics called attributes or 'commit features'. For example, code complexity metrics, code size metrics, code change metrics [Bird et al. 2009] and their corresponding values. These

attributes or features of the piece of code from a (.java file) are also called *Independent Variables* – which are the input to the algorithms used to perform prediction. There are various metrics developed with properties at different granularity levels; class, method, process and so on [Catal and Diri 2009].

Metric tools (Jhawk[1], locmetrics[2]) are used to calculate these metrics. Some of these tools have been found to produce different metric values when used to calculate the same type of metric [Lincke et al. 2008]. Tool inconsistencies could threaten the validity of the defect data. When tools are not shared and are likely to calculate the same metrics differently, it then makes it impossible to reproduce the results. For example, NASA metrics data sets have had erroneous values of 1.1 lines of codes for the LOC metric [Boetticher et al. 2008], that is a method having 1.1 lines, is not realistic. These errors could be from the metric tools they used and were not publicly available. In addition, the source code from which the metrics were derived is also not publicly available. Thus, the metrics are not reproducible and cannot be independently validated [Shepperd et al. 2013].

The SLR on finding the most effective metrics by [Radjenović, Heričko, Torkar, and Živkovič 2013] from 1990 - 2011, reported that object-oriented metrics such as Chidamber and Kemerer [1994] at class-level were mostly used among all metrics. Although Hall et al.'s SLR discovered that for higher model performance "*independent variables used by models performing well seem to be sets of metrics (e.g., combinations of process, product, and people-based metrics)*"[Hall et al. 2012]. As described in Kim et al. [2011] the *Dependent Variable* is the label of a predicted module, file, method or class. The label can be categorical values for binary classification; defective or not. The labels can also be continuous values for building regression models; that is the number of defects in a module.

Archive Stage: This stage is where metrics/defect data (uncleaned or cleaned) are stored and made publicly available for researchers to perform repeatable, refutable and verifiable experiments [Menzies

[1]http://www.virtualmachinery.com/jhawkprod.htm
[2]http://www.locmetrics.com

et al. 2007]. The PROMISE repository developed by Menzies et al. [2012][3] has been widely used and is helpful in making defect data sets from differents sources accessible to the public[4]. There is possible 'sampling bias' when data is contributed to a repository. Sampling bias occurs when a portion of the dataset in which the prediction models perform well are submitted to the repository, and the remaining portion of the data where models did not do well are not shared [Liebchen and Shepperd 2008]. A key issue with repositories is that when contributed datasets are easily accessible to users, it could be easier to use them for experiments without checking quality of the data (like NASA having 1.1 LOC for a module or duplicates [Gray et al. 2012]). In this case, defect data repositories have an important role to play in terms of data quality Shepperd et al. [2013], if not, producing unreliable studies may be propagated easily since the prepared data sets can be easily downloaded and used for experiments (chapter 4 provides more data on this problem).

2.2 Prediction modelling technique

Different studies use different classification methods. The most common are machine learning algorithms and statistical techniques [Malhotra 2015]. Examples of machine learning algorithms include Random Forests [Breiman 2001] and Support Vector Machine (SVM) [Gray et al. 2009]. A common statistical technique is Naïve Bayes by Thomas Bayes. The probability equation taken from [Rish 2001] can be expressed as follows:

$$P[\mathbf{F}|C] = \prod_{i=1}^{n} P[F_i|C] \tag{2.1}$$

$$P[C|\mathbf{F}] = \frac{P(\mathbf{F}|C)P(C)}{P(\mathbf{F})} \tag{2.2}$$

The definitions of the parameters in the equations are:

[3]now moved to https://zenodo.org/communities/seacraft Accessed: 12th Nov 2017
[4]the widely used datasets from NASA Metrics Data Program is available in PROMISE repository

$\mathbf{F} = (F_1, \ldots, F_n)$ is a feature vector

f is a feature in the feature vector (return type, parameter count)

C is a class (defective or not)

$P(C|\mathbf{F})$ is the posterior probability of a class (defective and not)

$P(C)$ is the prior probability of a class

$P(\mathbf{F}|C)$ is the likelihood of a feature to be defective or not

$P(\mathbf{F})$ is the prior probability of all features (it is the same when calculating likelihood of the classes)

Equation 2.2 works out the likelihood that a feature belongs to a defective or non defective class C. The mathematical algorithms associate the metrics values with the dependent variable. The dependent variable can be categorical, that is defective or not, or continuous, that is the number of defects.

I will give an example of categorical Naïve Bayes classifier Equation 2.1 because I use it in my work, see Table 2.1. It is naïve because it considers each variable as a separate contributor towards the dependent variable. For this discussion I will create data from my example code in Figure 2.4. The code is in Java where I attempt to predict whether it is defective by looking at the return type of the `writeResults()` method. I also look at the number of parameters in my method signature. I then add some modified methods I used in my experiments in chapter 5 for better illustration. None of these methods are defective, but I mark some of them as such for this example.

Naïve Bayes deals with variables that are categorical. As such, return type, parameter count, and defect count have to be transformed into binary form. This is possible by using thresholds for each variable. Return types that are void will be 'Yes', otherwise 'No'. Likewise, parameter count above 2 should be binary; that is a method with less than 2 parameters is a 'No' and above 2 is a 'Yes'. The same conversion is applied to defect count. Table 2.1 will then transform to Table 2.2 and to Table 2.3.

Table 2.1: Illustration data showing the methods and parameters. The independent variables are determined by number of parameters for each method and its return type. The dependent variable is the number of defects.

Method	Independent Variables		Dependent Variable
	Return Type	Parameter Count	Defect Count
writeResults()	void	0	0
findPlugin(String source, File folder)	void	2	1
parsePackage(Package p, PrintStream ps)	void	2	4
nextFile()	File	0	0
getOnlyJavaFiles()	List<File>	0	0
parseOnlyJavaFiles(Dir rdir, List javaFiles)	void	2	0

Table 2.2: Illustration data converted to binary data using different thresholds. The methods are numbered for readability. The independent and dependent variables are converted to binary using thresholds in each respective column.

	Independent Variables		Dependent Variable
Method	Return Type = void	Parameter Count >2	Defect Count >1
1	Yes	No	No
2	Yes	No	Yes
3	Yes	No	Yes
4	No	No	No
5	No	No	No
6	Yes	No	No

Table 2.3: Illustration data summarised based on the matching between independent variable values and those of the dependent variables values.

		Return Type = void			Parameter Count >2		
		Yes	No	Total	Yes	No	Total
defective	Yes	2	0	2	0	2	2
	No	2	2	4	0	4	4
		4	2	6	0	6	6

If a new method is added its prediction outcome will be based on probabilities calculated on the previous data. The likelihood of the two classes (defective and not defective), $P[\mathbf{F}|C]$, is first calculated for all features. Followed by the posteriori probabilities, $P[C|\mathbf{F}]$, of both classes as follows.

From equation (2.1) the likelihood (Not Defective | Defective) is,

$$P[\mathbf{F}|C] = \prod_{i=1}^{n} P[F_i|C]$$

Breaking equation (2.1) for all the two features in our data we now have,

$$P[\mathbf{F}|C] = \frac{P[\mathbf{F} = (f_1, f_2)|C = \text{not defective}]}{\text{all non defectives}} \tag{2.3}$$

$$P(C) = \frac{P(C = \text{not defective})}{\text{all methods}} \tag{2.4}$$

Substituting equations (2.3) and (2.4) into the numerators of equation (2.5),

$$P[C|\mathbf{F}] = \frac{P(\mathbf{F}|C)P(C)}{P(\mathbf{F})} \tag{2.5}$$

$$P[C = \text{not defective}|\mathbf{F}] = \left(\frac{2}{4} \times \frac{2}{4}\right) \times \left(\frac{4}{6}\right) = \frac{0.1667}{P(\mathbf{F})} \tag{2.6}$$

$$P[C = \text{defective}|\mathbf{F}] = \left(\frac{2}{2} \times \frac{2}{2}\right) \times \left(\frac{2}{6}\right) = \frac{0.3333}{P(\mathbf{F})} \tag{2.7}$$

P($\mathbf{F}$) is the prior probability of all features for defective and not defective instances,

$$P[C|\mathbf{F}] = \text{posterior probability(Not Defective)} = \frac{0.1667}{0.1667 + 0.3333} = 0.3334 \approx 33\% \tag{2.8}$$

$$P[C|\mathbf{F}] = \text{posterior probability(of Defective)} = \frac{0.3333}{0.1667 + 0.3333} = 0.6666 \approx 67\% \tag{2.9}$$

The result shows the posterior probabilities. Such that a new method with File return type and more than 2 parameters will be classified as defective. The classification is based on the maximum posteriori probability [Rish 2001], i.e, 67% probability calculated from the previous data.

2.3 Building the prediction model

Experiment Stage: Typically an algorithm is trained based on the defect dataset which is a combination of numerical values calculated from source code. The algorithms find patterns from combinations of these values and separate them into different classes. These patterns can be seen as if-then rules useful for classifying new patterns accurately [Quinlan 1993]. A piece of software can have multiple releases. A model can be trained on one release and used to make predictions on another release. With new releases come new changes. So, slow growth of data could affect the building of models. Defect prediction has been motivated towards building models in one organisation and used in a different organisation (cross company defect prediction) [Kitchenham et al. 2007]. Modelling is a complex process and mistakes can be made easily.

Gray et al. [2009] introduced an extensive framework for building reliable prediction models. Gray et al. [2009]'s techniques ensure there are no repeated instances in a training set and testing set. Otherwise, having duplicates in both sets nullifies the theory of prediction. The theory of prediction is that a trained model should make a prediction on code it has not seen before. Song et al. [2011] proposed a new framework by adding a metric (or attribute or feature) selection stage. In Figure 2.5 the study also added a retraining step - where a trained model is retrained on the whole training set after being trained on different partitions of the training set. The trained model is then used to predict what the labels are given unseen test data. How well the model performed is then evaluated using standard performance measures.

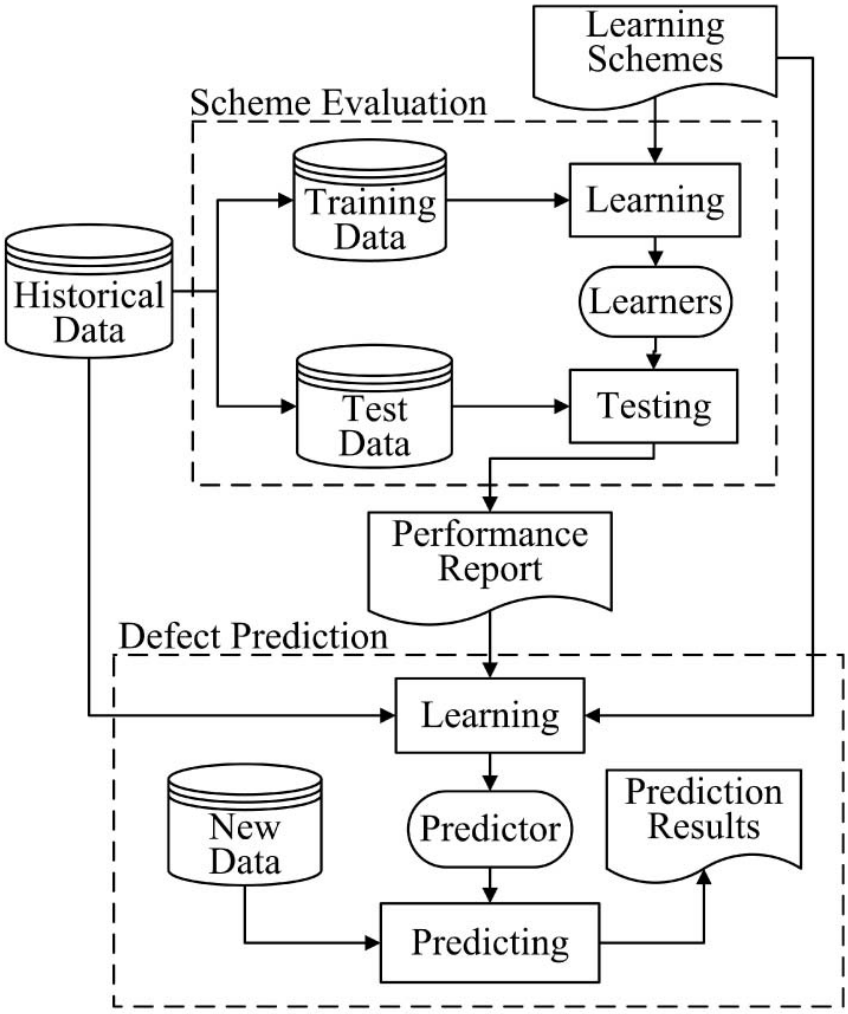


Figure 2.5: A procedure for evaluating a defect prediction model [Song et al. 2011]

I use some these classifiers which I elaborate in chapter 5. Some of the main challenges at this stage have to with data quality, the procedures followed in handling the datasets when building prediction models and how the performance of the models are measured.

Data handling: There are many challenges within defect datasets. Some source codes can have few defects in them, and training classifiers on such imbalanced data makes the algorithm struggle to learn from few samples that are not sufficient to allow the classifier separate new patterns. This could make

the classifier's performance poor. There are many techniques to handle data imbalance, for example, by increasing the number of defective patterns or instances in the dataset so that the classifier would have enough training samples to make better predictions. Many of these techniques have been compared by Chawla [2003] and recently by Rodriguez et al. [2014]; including cost sensitive, sampling and ensemble methods. Additionally, how data is split for training and testing can be flawed, this is where novelty and replication problems occur [Fokkens et al. 2013]. For example, Rodriguez et al. [2014] uses 5 by 5 cross validation when 10 by 10 is recommended. The latter provides more reliable estimates on how the model may perform when given new datasets to classify.

Since the idea of cross-validation is to split the dataset into (n) approximate folds so that in each run the model can be trained on (n-1) folds and tested on the one fold. The procedure can be run 10 times, each time randomising the instances so that the model is not used to the same patterns most of the time. Fewer folds for training and testing, and less randomised instances could bias (over fit) the classifier to that dataset and the model could perform poorly on new unseen data [Hand et al. 2001]. The problem is even if standard techniques are used to develop prediction models, the results could remain unreliable as long as stages 2, 3 and 4 are not addressed.

2.4 Evaluating the performance of the prediction model

When the model predicts whether a piece of code is defective or not, its performance is measured using different approaches. Song et al. [2011] elaborate on the evaluation process in Figure 2.6.

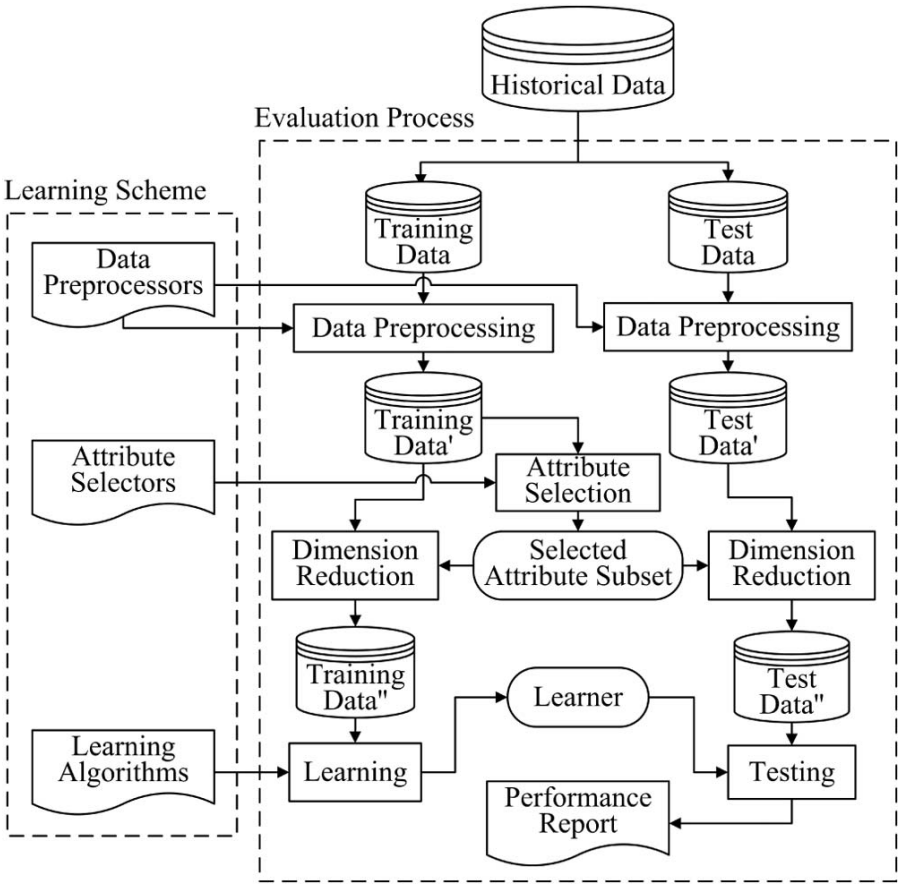


Figure 2.6: A procedure for building a defect prediction model [Song et al. 2011]

Again, prediction performance measures that partially explain model's performance are used, e.g. recall only or precision (Table 2.5). Ostrand and Weyuker [2007] investigate the success of prediction models reporting common measures. The paper concluded that combining measures better explain performance of prediction models. Furthermore the combination of what percentage of defects were missed [5] and percentage of defective files/modules depends on interest. These two measures are important to know effectiveness of the prediction. Precision, Recall, Type I, Type II have all been studied. Type I and Precision focus on false positives (classified non defectives as defectives); Type I is more relevant in terms of prediction. Comparing Type II and Recall, Type II was selected as more relevant and both focus on False Negatives (classifying defectives as non defectives is most severe as it leads to failure

[5]Type II misclassification rate or False Negative rate

in operation) Table 2.5. However, a comprehensive measure Mathews Correlation Coefficient (MCC) which tests the model in all four parts of the confusion matrix Table 2.4 is recommended [Hall and Bowes 2012]. A prediction model that is not evaluated in full may not be useful for practitioners. When a model misses a defect and that defect leads to failure in operation, there is a huge cost to the company and the user.

Table 2.4: A Binary Confusion Matrix

	Observed Positive	Observed Negative
Predicted Positive	True Positives (TP)	False Positives (FP)
Predicted Negative	False Negatives (FN)	True Negatives (TN)

Table 2.5: Compound Performance Measures from a Binary Confusion Matrix

Measures	Defined As	Meaning
Recall	$\frac{TP}{TP+FN}$	Proportion of actual positives found.
Precision	$\frac{TP}{TP+FP}$	Proportion of predicted positives which are true positives
$F-Measure$	$\frac{2\times Recall\times Precision}{Recall+Precision}=\frac{2TP}{2TP+FP+FN}$	The harmonic mean of *Precision* and *Recall*.
Matthews Correlation Coefficient (MCC)	$\frac{TP\times TN-FP\times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$	A regression coefficient using all four quadrants of a confusion matrix.

2.5 Summary of defect prediction: 5 Stages

Based on the defect prediction literature, the process can be summarised into 5 important stages, see Table 2.6. The second column lists all stages of the defect prediction process. The description of what is at each stage is given in the third column.

Table 2.6: Software Defect Prediction process and its Defect Data Life-Cycle

Sn	**Stages**	**Description**
1	Creation	Developers write code, Defects are labelled, fixed, inserted, reported
2	Linkage	Defect ID in bug DB is matched with fix point in version control
3	Metrics	Metrics are calculated from features of code like line of code counts
4	Archive	Metrics called data are stored for public to access via website
5	Experiment	Researchers, practitioners, reviewers use data to build and test prediction models

In light of all this complex process of building prediction models, there are potential threats to validity at every stage of the 5 defect prediction process stages Table 2.6. Therefore, repeating an experiment to get the same result is a challenge. Especially when the original studies are not rigorously performed or do not report sufficient information. When a study is not reproducible or replicable how can it be trusted? This takes us to the next section which elaborates on the importance of having replicable and reproducible studies and the difference between the two terms.

2.6 Defining the terms replicability and reproducibility

Thomas Kuhn says *'normal science' means research firmly based upon one or more past scientific achievements, achievements that some particular scientific community acknowledges for a time as supplying the foundation for its further practice... and "a paradigm is an accepted model or pattern"* [Kuhn 1963]. In order to shift from a paradigm and drive progress scientific rigour must be applied. Firm consensus must be reached from one paradigm to another [Kuhn 1963]. Therefore scientific methodology requires us to repeat experiments to check that other people can get the same results. If people cannot get the same results, the original study may not be valid. In the previous replication examples given in chapter 1 (Marangos [2000] particles of light, Fleischmann et al. [1989] cold fusion) one can repeat an experiment to get the same or different findings. How about in this age of computing with resources being accessible? How can these concepts be distinguished? The terms replication and reproducibility are often interchanged, but they carry different meaning. Each term determines how experiments are done and the expected outcomes and their complexity. Replication means to repeat an experiment by independent researchers within a different environment, with changes to the original study aimed at getting consistent results. Reproduction is to recompile the same artefacts used for a study, including data, analysis and procedures for validation [Leek and Peng 2015, Madeyski and Kitchenham 2017] to get the same results. I provide the details and history of both concepts in the following sections.

2.6.1 Origin of Replication (Robert Boyle, 1600s)

[Gandrud 2013] talked about Francis Bacon's book 'Colours of Good and Evil' [Bacon 1890] originally published in 1605. That replication is referred to as demarcating science from non-science and science is a systematic way of collecting knowledge into testable claims. A famous experiment that sparked considerable debate about replication: Robert Boyle's New Experiments Physico-Mechanical, Touching the Spring of the Air, and its Effects. The book published by H. Hall, printer to Oxford University, was

about the first controlled experiment that measured the effect of lowering air pressure in an air pump (of ~28 litres and ~38cm diameter in size, see Figure 2.7) [West 2005]. Shapin et al. explained this debate that at that time Boyle's claims about his observations "*could be turned into matters of fact by replication of the pump*"[Shapin et al. 1985]. The moment Boyle's book on New experiments was published, within a year 1660 to 1661 August, other researchers attempted to replicate the pump but failed. It took an eye witness Huygens present at the air pump trials to be able to replicate it. The replication took place at a different location with some changes. The research groups in Germany and Florence had the full text but were unable to replicate the pump. Hobbes published (in Dialogus physicus) some of the errors in the pump. An example of one error is air leaked between sucker and cylinder. Once the flaws were identified the custodian of the pump Royal society requested Boyle to make corrections within a year [Shapin et al. 1985].

2.6.2 Importance of Replicability

An important lesson here is that within a year (1660 - 1661) researchers set to replicate the air pump and found errors. The custodians Royal Society requested the author Boyle to make corrections to existing pump immediately. The speed at which findings are replicated in order to be verified is important to minimise error propagation among the scientific space. The experiment would not be accepted as fact until it was successfully replicated. Indicating the significance of replication as the scientific method to establish facts. Which means that quantifying replications in any scientific field is crucial to evaluate whether some facts have been established to reduce false claims and correct errors as in the case of Robert Boyle. West [2005] further reported that Boyle and Hooke extended the work on effects of low pressure, on magnetism, sound propagation, behaviour of a pendulum etc. [West 2005].

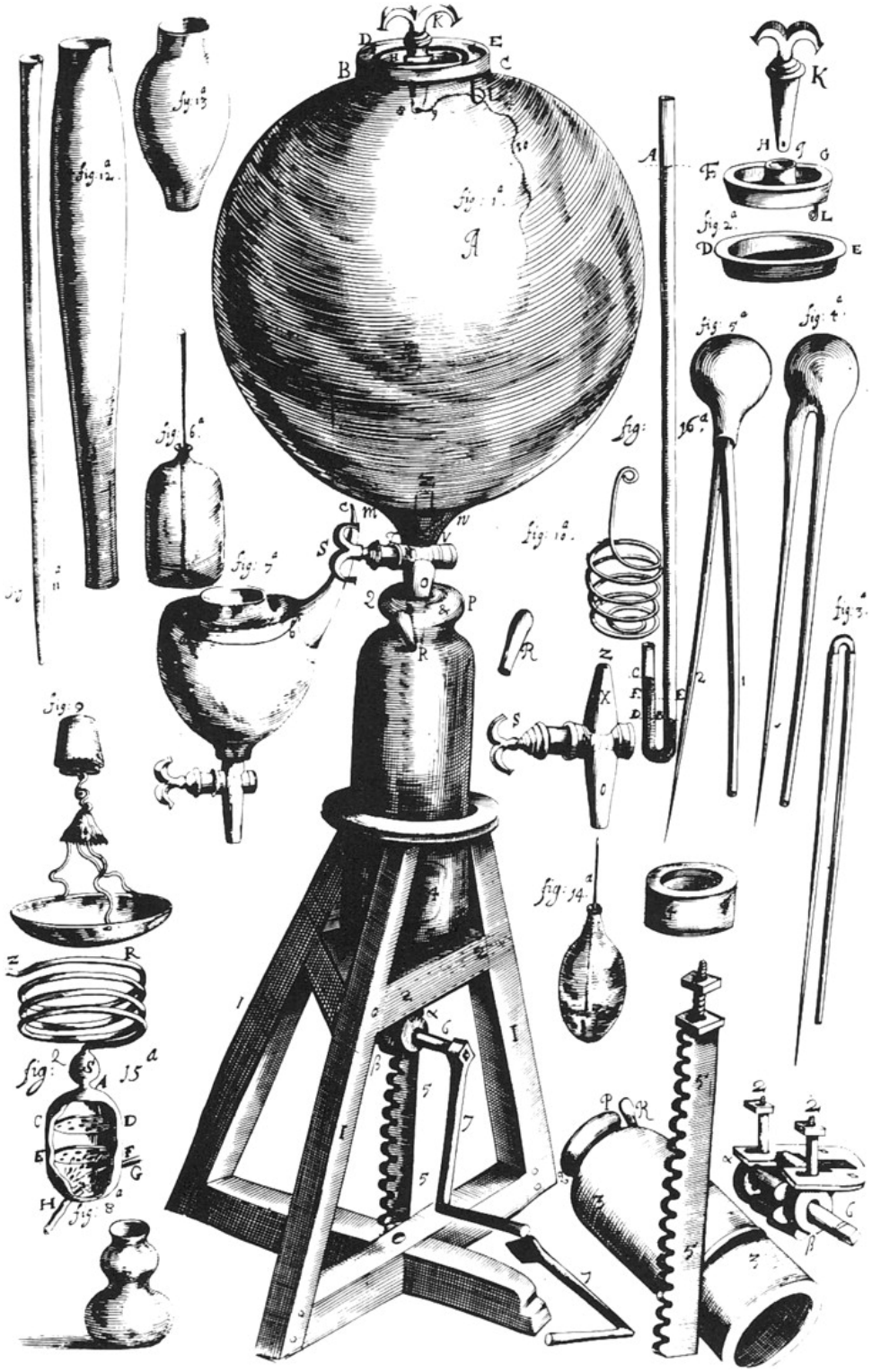


Figure 2.7: *"Engraving of the air pump devised by Robert Boyle and Robert Hooke (1635 - 1703). The complete pump is shown at center, and some of the disassembled parts are at right. Various small pieces of equipment that were used in the experiments are also shown"* West [2005]

2.6.3 Origin of Reproducible Research (Jon Claerbout, 1990)

More recently with the advent of technology, computing and software engineering replication could branch out to reproduction. It is because one can access the exact same materials used for an exeperiment through repositiories, file transfers and so on. Unlike in replications some materials are not accessible. For example the same composition of glass, leather used for Boyle's pump might have intrinsic difference to those in a different environment. Madeyski and Kitchenham [2017] summarised the origin of reproducible research documents in their paper on adopting reproducible research in Software Engineering. The paper reported that Gandrud [2013] attributes the term "reproducible research" to Professor Claerbout (Stanford University) in 1990. Claerbout ensured makefiles[6] were created to compile all results of the studies including figures during the Stanford Exploration Project.

2.6.4 Importance of Reproducibility

Gandrud [2013]'s book further explained that accessing the three parts of a study, text, data and analytic procedures (algorithms) achieves reproducibility. Some of the advantages are that: research can be verified, new work can be created, it allows better teamwork, replication is easier by doing re-analysis, and lowers the effort of doing what has been done already. Conversely, producing non-repeatable research, even by oneself [Baker 2016] is a waste of time, effort and research funding [Ioannidis 2017]. Failed reproducibility may disrupt advancement of research [Shepperd et al. 2014, Ioannidis 2017].

2.6.5 Contrasting the Terminologies

Given the definitions of replication and reproduction and their examples the two terms differ. Reproduction applies when a researcher has access to the same materials, tools and is able to repeat the study and get the same result (in the case of makefiles). Replication applies when a researcher uses differ-

[6] a file kept in a directory containing some executable commands called by the keyword 'make'

ent materials and possibly tools to get similar behaviour of a system (in the case of Boyle's air pump). As a result of this difference these two concepts may also be measured differently depending on the context. For instance, in the context of defect prediction, using the same tools and methods the measurement/independent variable (e.g. model performance) outcome of the original and its reproduction study might be expected to be the same. Perhaps with some differences due to rounding errors. Using different materials and environment the expected outcome would not be as straight forward. There could be many variations intrinsic to the contextual factors that differ from those of the original environment and materials.

2.7 Adopting the Terminologies to Defect Prediction

Previous work has looked at the reproducibility of data mining studies [González-Barahona and Robles 2012]. Defect prediction studies invariably are based on data mining. González-Barahona and Robles [2012] propose a process model to gauge the reproducibility of data mining studies by identifying key elements of the research including: data source, retrieval methodology, raw dataset, extraction methodology, study parameters, processed dataset, analysis methodology, and results dataset. Goodman et al. [2016] suggest that in any scientific field the kind of replication must be clearly specified, including the components of studies being replicated with respect to the knowledge derived and the limitations of using statistical significance as the only basis for conclusions [Goodman et al. 2016]. In this dissertation part of the Gómez et al. [2014] replication taxonomy is adopted; it tracks changes made to components of an original study, and identifies the different types of replications that can be performed. The taxonomy was originally defined for software engineering human-centric experiments, but I adapt it to defect prediction experiments (chapter 3 presents the adaption).

According to the taxonomy of Gómez et al. [2014], replication in software engineering can be categorised into three broad types (see Table 2.7). *Literal* is a type of replication done by authors of the

Table 2.7: The identified replications in this study that are mapped & tagged to the considered categories of the Gómez et al. [2014] replication taxonomy.

Replication type	Protocol	Operationalisation	Populations	Experimenters	Replication name & changed (Δ) components
Literal	=	=	=	=	Repetition
Operational	=	=	=	≠	Δ-experimenter (A)
	=	=	≠	=	Δ-populations
	=	=	≠	≠	Δ-populations/-experimenter (B)
	=	≠	=	=	Δ-operationalisation
	=	≠	=	≠	Δ-operationalisation/-experimenters (C)
	=	≠	≠	=	Δ-operationalisation/-populations
	=	≠	≠	≠	Δ-operationalisation/-populations/-experimenters (D)
	≠	=	=	=	Δ-protocol
	≠	=	=	≠	Δ-protocol/-experimenters (E)
	≠	=	≠	=	Δ-protocol/-populations
	≠	=	≠	≠	Δ-protocol/-populations/-experimenters (F)
	≠	≠	=	=	Δ-protocol/-operationalisation
	≠	≠	=	≠	Δ-protocol/-operationalisation/-experimenters (G)
	≠	≠	≠	=	Δ-protocol/-operationalisation/-populations
	≠	≠	≠	≠	Δ-protocol/-operationalisation/-populations/-experimenters (H)
Conceptual	Unknown	Unknown	Unknown	Unknown	(only hypotheses are retained)

original study. In effect, this type of replication is named Repetition because no component of the original study is changed; the same experiment is run by the same authors using the exact tools on the same data to avoid bias in the results. Modifying any component of the original study changes the type of replication to *Operational*. For example, if different authors replicate an original study while data and tools remain the same, it is the *Operational* replication type with Changed-experimenter (in effect the same as reproduction); experimenter is the component. Under the Operational replication, 15 changes can be made to the original study, and each change is given the appropriate name to reflect the change (Table 2.7 column on the right identifies these changes) for example the populations being studied may change. The third replication type is called *Conceptual* because every aspect of the original study is changed except the hypotheses. Applying this taxonomy to new and existing replications is crucial in aggregating replication types and results, to consolidate and synthesise new knowledge.

2.8 Summary

Within the 5 stages of defect prediction process I have highlighted the challenges faced. Researchers may face impediments when verifying studies through replication and reproduction. These two concepts vary in terms of their approach to the experiments and expected results. Reproducibility problems have been highlighted in other fields, for example, a recent study in Nature Epistemology suggests that many

researchers fail to reproduce other researchers' work and this failure includes their own work [Baker 2016]. In Cognitive Science reproducibility is challenging due to the implementation of computational models in computer code [Lane and Gobet 2003]. In light of the current problems within the defect prediction process this dissertation aims to identify the number of replicated original defect prediction studies. I also aim to statistically identify characteristics of these studies that are likely to relate to a paper being replicated. In the next chapter I detail my methodology on finding replication studies and measuring their outcomes. I further elaborate on the approach I use to reproduce these studies and statistically confirm some of the factors that have impact on prediction replication and reproduction.

3. METHODOLOGY

In this chapter, I propose a research validation process to provide new evidence on the state of replication in defect prediction. I also highlight technical details affecting validity assessments through reproducibility. The methodology is broken down into two approaches, theoretical (qualitative) and empirical (quantitative) approaches. The theoretical approach (detailed in chapter 4) explains the systematic review and analysis of replication studies. In the empirical approach (detailed in chapter 5) the experiments undertaken to test the reproducibility of research are explained. Investigations into the reproducibility failures encountered are also explained (detailed in chapter 6). In this chapter these explanations are supported with a mapping of how both approaches fit together.

3.1 Research Method

In Empirical Software Engineering there are different research methods [Runeson and Höst 2009, Vegas 2015]:

- survey is collecting information from a sample, not only by questionnaire or interview
- case study is an enquiry on one phenomenon based on a variety of sources describing that phenomenon in real life
- controlled experiments measure a variable and observe its effects on another
- quasi experiment is similar to a controlled experiment except that there is no random assignment to control groups

As described by Vegas [2015], over the past 20 years an Empirical SE PhD can generally fall under two categories either of which can use any of the research methods. The first category gathers knowledge about a specific topic, for example Seaman [1998], through several experiments, characterised aspects of communications between team members when developing software. Shull and Basili [1998] ran several experiments and synthesised knowledge about different reading techniques to inspect code.

The second category proposes a methodological advancement of Empirical SE research: For example, Daly [1996] proposed a multi-method technique to conduct empirical research combined with a replication strategy to achieve reliability and generalisability of findings; Solari [2013] identified the appropriate contents to include in a replication package; Ciolkowski [2011] proposed a methodology for quantitatively aggregating evidence from controlled experiments; Jedlitschka [2009] proposed a method of reporting useful information for managers to make decisions; and Gómez et al. [2014] proposed a replication taxonomy which organises the order in which types of replications should be done.

I use the Empirical SE controlled experiment research method which is defined as an attempt to combine a set of components into a defined protocol (framework) whose outcome (dependent variable) can be measured [Runeson and Höst 2009, Vegas 2015]. My thesis aims to satisfy this approach by proposing a methodology to advance replications in defect prediction. I support this thesis by identifying replications of 208 controlled experiments from an existing SLR [Hall et al. 2012]. I also assess the confirmations of the findings (agreements) followed by a quantitative analysis of factors associated with replications. Finally, I do a reproducibility check on those replicated studies and identify potential factors causing reproducibility failures.

3.2 Theoretical approach

Under this part of the methodology there are six stages. Each stage is broken down into multiple processes. Each process passes information sequentially to its dependent process. The stages begin from

the top right of Figure 3.1 and progress downward. The same order is followed on the left side of the figure to complete the review. Systematic reviews are an important aspect of gathering evidence from the literature to answer research questions and unify research goals: This type of review is widely used in Software Engineering (SE) [Kitchenham 2004, Kitchenham et al. 2009, Catal and Diri 2009, Hall et al. 2012, Radjenović et al. 2013, Wohlin 2014, Silva et al. 2014, Malhotra 2015, Wahono 2015, Hosseini et al. 2017]. Therefore I use a systematic review to find valuable insights about replication in defect prediction.

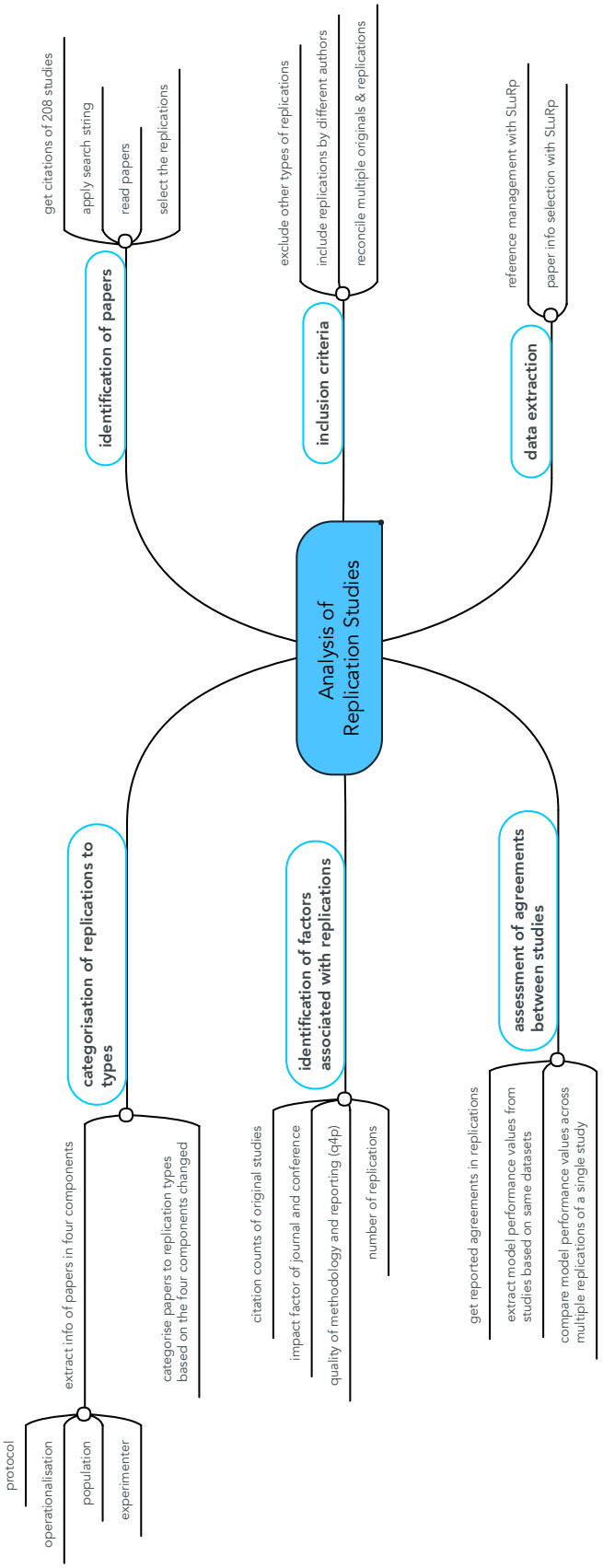


Figure 3.1: The six stages and their processes used in the analysis of replication studies

3.2.1 A Qualitative and Quantitative Analysis of Replication Studies in Defect Prediction

`Stage 1 - Identification of replication papers:` In the first stage, I aim to find studies that conduct one or more replications. The number of studies found will answer (RQ1). To replicate a study, that replication must have cited the original paper. Therefore, the first place to begin the search is citations of an original study. The criteria and search strategy used to develop the systematic review process will be synthesised from [Wohlin 2014, Hall et al. 2012, Silva et al. 2014] and adapted for finding replication studies as follows:

1. Select a representative base set of original defect prediction studies [Wohlin 2014]: so the 208 original defect prediction studies from [Hall et al. 2012] will be used.
2. Search through references that cite the original study (forward-snowballing) [Wohlin 2014], between 2000 - 2017, because the original paper must have been published before the replication was conducted [Silva et al. 2014] (this means that references within the original studies will not be searched).
3. Use a reliable search engine (e.g. Google Scholar) to ensure sufficient coverage of papers that cite original studies. On the *'cited by #papers'* page of each paper the ~Replicate OR ~Replication OR ~Replicated string and selected *'search within citing articles'* filter will be used. In effect, only papers that used these terms or their synonyms (denoted by tilde (~)) will be returned. Applying this technique will reduce the number of papers to be assessed as replications and reduces false positives. From the returned results page, we will read the paper title and its summarised phrases to identify if the paper was a replication of an original study in the 208. If not sufficient, the whole document will be accessed to find the context in which the term was used, as suggested by Wohlin [2014]. For this search the in-built search feature of the web browser or document reader will be used to find the term replication. If the replication term is not in the document the paper will be

read in full to determine if it is a replication. Using this approach will help identify a set of papers that replicated a sub-set of the 208 original studies.

4. Use SLuRp [Bowes et al. 2012], an open source SLR tool to aid the review process for effective tracking, storing, and retrieving information [Hall et al. 2012], and to achieve reproducibility for reviewers, researchers, practitioners to repeat the study and be able to ascertain its credibility.

`Stage 2 - Inclusion criteria:` In the second stage, I aim to exclude internal replications (sometimes known as self replications). There are two main types of replication. The replications conducted by the original authors themselves and those replications done by authors other than the original authors. These two can be called internal and external replication [Silva et al. 2014]. The latter is the preferred type [Silva et al. 2014, Gómez et al. 2014, Fucci et al. 2016] to minimise threats to the validity of a study. Since bias introduced by original authors (as discussed in chapter 1) has been shown to be the dominant factor impacting model performance, and consequently has the tendency to disrupt replication and hinder progress in a field [Shepperd et al. 2014]. I will exclude internal replications or extended work by the original author(s). If the author(s) have extended an original piece of work, I will consider this work to be one paper, and any replication of either of these two is a replication of an original study. I will consider any author (whether a lead author or not) to be an author of the paper. Consequently I will be able to track all replications by all authors of original studies and exclude them.

`Stage 3 - Data extraction:` In the third stage, data from the identified replication studies will be extracted and validated amongst authors. The approach is adopted from [Hall et al. 2012] because the study is one of the *"very prominent 'gold sets' as published SLRs"* and *"define their work in enough details for us to construct data sets for simulations"* [Yu et al. 2016], making it a standard followed in this dissertation (a recent SLR [Hosseini et al. 2017] also used the same extraction technique).

1. `Tool for extraction:` For reproducibility I use the SLuRp tool [Bowes et al. 2012]. SLuRp is a web-based tool developed to make Systematic Literature Reviews (SLR) reproducible and also provides effective information storage and retrieval. SLuRp has been assessed as the best out of four SLR tools in SE Marshall et al. [2014]. I will not use all of SLuRp's functionality, many more useful SLR management features are described in [Bowes et al. 2012]. I provide the following steps as a summary of SLuRp together with how I will use it for data extraction.
   (a) Import BibTeX files and store references to all original and replicated studies.
   (b) Assign more than one researcher (in my case two researchers are assigned, Mahmood and Bowes) to independently store extracted information from each paper.
   (c) Allow the researchers to modify and approve extracted information.
   (d) Disagreements between the researchers are flagged by SLuRp.
   (e) Create forms based on contextual and methodological information that must be extracted from each paper.
   (f) Store extracted information in the SLuRp database.
   (g) Retrieve stored information using SQL queries and organise into result tables.
   (h) Export tables as LaTeX tables. Graphs and box plots are available.
   (i) Edit entire final report with SLuRp LaTeX editor, including results, tables and compiled to produce the final paper - this functionality allows researchers produce reproducible research work.
2. `Reference management` References of the set of studies will be stored in a BibTeX file. Original studies will be labelled in BibTeX using the format [paperID#] and replication studies as [#paperID#]. Because some original studies may be replicated more than once; this format is crucial to track the number of replications per original study. For example, an original [pid1] will have one replication as [1pid1] within the bibliography database.

3. `Extraction of selected data from final set`

   Three sets of data will be extracted to allow RQ2, RQ3 and RQ4 to be answered. The first set of extracted data (for RQ2) characterises how defect prediction studies are performed. The dataset is based on the defect prediction characteristics presented in Hall et al. [2012] and Hall and Bowes [2012]. These characteristics include:

   (a) dependent variables

   (b) independent variables

   (c) algorithms

   (d) dataset

   (e) tuning

   (f) cross validation

   (g) statistical analysis

   This dataset of defect prediction characteristics can be analysed for insights on replication practice and to categorise replications based on changes replications make to the original studies in terms of these characteristics. The information we collect allows us to categorise replications into their respective categories (as defined in Table 2.7).

   Once the final set of papers is identified, 5 papers will be read independently between two authors (Mahmood and Bowes, by way of a validation check on the data extraction process) and their data extracted, while agreements will be reached on this data extraction using SLuRp to minimise threats to validity. The review is a significant step to reduce errors or bias that can be introduced by researchers, and in turn to reach a sound conclusion. Information on the remaining papers will then be extracted by me. The second set of data extracted (for RQ3) allows us to determine which features of defect prediction studies make it likely an original study will be replicated. This set of data is: study quality, publication venue, citation count and dataset (as explained in chapter 2). The third set of data to be extracted (for RQ4) allows us to establish whether the results

of a replication study are comparable to the original. Chapter4 describes the process by which I establish and measure agreements between a replication and the replicated original results.

`Stage 4 - Categorisation of replications into types:` In the fourth stage, the process identifies how replications are performed by applying Gómez et al. [2014]'s replication taxonomy to categorise the identified replication studies into types. The extracted information from the papers initially stored in SLuRp will be retrieved as tables using the SQL functionality. The information will then be divided into four components based on Gómez et al. taxonomy. The retrieved information from SLuRp will be presented in these four components, adapted from Gómez et al. [2014], see Table 3.1. Each component changed may assist in the discovery of unknown factors that affect replication results. I detail the four components of a study as follows;

Protocol is the overall study design. In defect prediction the framework that pulls together different sub-components to build a prediction system is the overall study design (protocol). Table A.1 shows the protocol sub-components I will use:

1. cross validation scheme used
2. whether parameter tuning was performed
3. which statistics were used to compare performance results
4. whether data cleaning was used

These factors are motivated by Hall et al. [2012] and Hall and Bowes [2012] as outlined previously in chapter 2. The protocol is the design before it is implemented (i.e. operationalised).

Operationalisation has two aspects, cause and effect. The cause is the process of implementing the protocol and considers the implementation environment. We consider the following implementation factors (again motivated by Hall et al. [2012] and Hall and Bowes [2012]):

Table 3.1: Changeable Components of Defect Prediction Studies adapted from Gómez et al. [2014]

Protocol	Operationalisation	Population	Experimenters
Definition: The configuration of subcomponents to observe an outcome **Changes:** -Experimental design of how treatments are allocated e.g. model building framework configures data preprocessing, parameter optimisation, cross validation, prediction, data collection framework configures defect linking, extracting and labelling -Statistical analyses	**Definition:** Mode of applying treatments (techniques) e.g training a model on train and test set gives unrealistic results than on train set only **Changes (cause construct: cause of differences in results):** -Literature sources, training, instructions for applying a procedure during experiments -Tools used for running experiments e.g IDE -Algorithms for, building prediction models, dealing with imbalance, linking defects **Changes (effect construct: effect on results):** -Defining dependent/ independent variables e.g. number of defects post release/code complexity -Process of calculating the variables e.g. number of defects fixed after released to customers and linked to the point the defect was introduced -Measuring model performance with different measures	**Definition:** The subject and objects properties used in a controlled experiment **Changes:** -Source code of project (open (Eclipse) or closed (NASA) source) -Design documents, programming language, size, complexity, maturity (years used and growth), domain etc. -Granularity of independent variables (metrics) such as class or method level, granularity of dependent variables such as defective or not and number of defects	**Definition:** The designer, trainer, monitor, measurer and analyst involved in the experiments (authors). **Changes:** -Different authors may or may not optimise parameters of an algorithm on the same dataset

1. Tools used
2. Algorithms used
3. Independent variables used

Algorithms have are available in data mining tools like Weka [Hall et al. 2009], in effect the tools help to implement a prediction framework. Therefore such tools and their versions must be considered because variations may cause differences in replication or reproduction results. Effect is the process of determining and defining the aspects of a model to be measured and selecting the appropriate measure. Since performance measures already exist (e.g. recall; measures the proportion of actual defects a model correctly predicted, see chapter 2), it is a question of 'which' measure appropriately measures the effect of the treatments on the model's prediction outcome.

Population is based on the systems analysed in studies. These systems are then mined from source code repositories (open or closed sources). Below are the population factors that are considered:

1. data source (which repository, open source could be Eclipse, Mozilla)
2. domain
3. language
4. granularity of defect data

The granularity i.e. method or class level where the defective or non-defective data are gathered is also part of this. The programming languages used, size of project (KLOC), maturity (years of use and development) etc. Changing any of these sub-components affects the population and likely the replication results.

Experimenters are the researchers that conducted the study.

As explained in chapter 2, Gómez et al. [2014]'s idea of the types of replication is based on changes to the subcomponents within these four components. Each component changed represents a type of replication. There are three broad replication types based on changes to the study components: i) literal (no change), ii) operational (15 changes), and iii) conceptual (only hypotheses of the original study are retained). The results of applying this taxonomy to the identified replication studies are provided in chapter 4.

`Stage 5 - Identification of Factors Associated with Replications:` In the fifth stage, for all 208 papers, as discussed previously we (Mahmood and Bowes) will extract the following factors to find out if any of the them has a relationship with the number of subsequent replications:

1. $quality_{4p}$
2. number of citations of a paper
3. publication venue
4. publication venue's impact factor
5. data sharing/availability

The focus on quality is necesary because Aksnes [2003] deems quality as the core knowledge that leads to further developments by other researchers, with lasting significance. I also focus on publication venue and study influence as Garousi and Fernandes [2016] report that highly cited papers make studies influential. Only the quality characteristic is not directly measurable. The $quality_{4p}$ assessment outcomes will be extracted using Hall *et al.'s* quality check for defect prediction studies [Hall et al. 2012] for the 208 original studies that have been replicated (see Table 3.2 for a summary of the four phased quality check criteria, $quality_{4p}$). The $quality_{4p}$ process assesses defect prediction studies in terms of whether they employ a reliable *methodological* approach to building prediction models and whether studies *report* sufficient information to comprehend a study.

Table 3.2: Summarised $Quality_{4p}$ Criteria by Hall and Bowes [2012][1,2] from Hall et al. [2012]

$Quality_{4p}$ Assessment Phases	Details of Phases
phase 1: Establishing that the study is a prediction study.	-Is a prediction model reported? -Is the prediction model tested on unseen data?
phase 2: Ensuring sufficient contextual information is reported.	-Is the source of data reported? -Is the maturity of data reported? -Is the size of data reported? -Is the application domain of data reported? -Is the programming language of the data reported?
phase 3: Establishing that sufficient model building information is reported	-Are the independent and dependent variables clearly reported? -Is the granularity of the dependent variables reported? -Are the modelling techniques used reported?
phase 4: Checking the model building data	-Is the fault data acquisition process described? -Is the independent variables data acquisition process described? -Is the faulty and non-faulty balance of data reported?

[1]Phase 1 assesses defect prediction methodological approaches [2]Phases 2, 3 and 4 assess reporting of prediction studies

$Quality_{4p}$ overlaps extensively with González-Barahona and Robles [2012]s' reproducibility criteria which includes checking the: data source, retrieval method, raw data, extraction method, study parameters, analysis method, results method, identification and description. Two elements of González-Barahona and Robles [2012]s' reproducibility criteria are missing from $quality_{4p}$ and these are data availability and data flexibility. Availability of data will be collected, i.e., an element's tendency to exist in the future. All the links of each study will be checked to confirm if data are accessible. In addition I will collect González-Barahona and Robles [2012]s' flexibility criteria, adaptability to different environments, by extracting the formats of shared data in terms of e.g. csv, arff etc. For open or closed source code repositories, metrics (e.g. object oriented metrics calculated on defective/non-defective code) can be collected to form defect data used as input for building prediction models [D'Ambros et al. 2010][Zimmermann and Nagappan 2008][Zimmermann et al. 2007]. For example the NASA MDP program provided defect datasets calculated from the raw source code of critical systems (e.g. Flight and

Satellite systems). The raw source code, being proprietary, was not available. However, it is possible to reproduce a study based on the defect data which was shared even though it was generated from a closed source.

Impact factor values of the publication venues the papers are published in will be collected from Journal Metrics (details are in chapter 4 in Table 4.9). I will also use the Source Normalised Impact Average (SNIPA) [Moed 2010] values which are based on the average citation per paper of a journal in that subject area. In addition, the ratings of journal/conference venues from Excellence in Research for Australia (ERA) will be extracted. In 2009, the Australian Research Council consulted the public, expert reviewers and academic bodies to rank journals and conferences, and produced the ERA rankings. The ERA 2010 rankings are used since other ranking bodies only provide journal impact factors and omit any ranking of conferences. ERA has 5 ranks according to research quality, see Table 3.3.

Table 3.3: ERA Ranking Categories

Rankings	Description
A*	flagship conference, a leading venue in a discipline area
A	excellent conference, and highly respected in a discipline area
B	good conference, and well regarded in a discipline area
C	other ranked conference venues that meet minimum standards
Unranked	A conference for which no ranking decision has been made

http://www.core.edu.au/conference-portal

`Stage 6 - Assessment of Agreements between Studies:` This stage assesses the outcome of the identified replication studies, whether they succeed or fail in replicating the original studies. Such information is useful to gauge the reliability of existing studies. If an original study is used by a different research group in a different environment (organisation) and is able to confirm the findings, then this shows the reliability of the initial hypotheses being tested.

Confirmed findings may be depended on by other researchers and practitioners with more confidence. Agreements reported in those studies will be recorded as Yes or No. Where both the original study

and its replication are based on the same data, the prediction model performances will be extracted and compared to confirm the agreements.

An important step towards establishing more reliable agreements is where a study reports confirmation but the values show a significant difference; such a wide gap may be considered an agreement for replications, but should not be considered as an agreement for reproduction. For example, an original study A that reports a prediction model identifying 80% of the modules as containing defects in a system (i.e. recall), while its replication B reports a recall of 70%, shows a 10% difference between the studies. The difference may end up being the defective modules that could cause system failure, although many studies would report an agreement of A and B. Fenton and Ohlsson [2000] report that a small number of defective modules lead to most failures of the system in operation. It is important to measure agreements by their model performance measure as small differences may have significant consequences.

3.3 Empirical approach

3.3.1 Reproducibility Checks Performed on Replicated Papers

Reproducibility checks that investigate the credibility of previous work are conducted in this part of the methodology. Reproduction is to repeat a study using the same materials with the aim of getting the same results as the original study to ascertain its credibility [Leek and Peng 2015, Madeyski and Kitchenham 2017]. Ioannidis [2017] suggests that *"reproducibility checks performed to date are still relatively few and include few replications each. It would be useful to understand which disciplines have high consistency in their results, which show high heterogeneity, and which have consistently nonreplicated results."* Influential papers are primary targets of reproduction because other researchers rely on them significantly [Ioannidis 2017] and, if not verified, such studies could misrepresent reality [Errington et al. 2014]. Studies need to be reproducible and replicable, and their multiple replications help to *"validate the validator"* [Miller 2005]: This mitigates replication bias (i.e. introducing bias that ends up invalidating an original study that is valid) [Ioannidis 2005]. Multiple replications can be combined to produce new knowledge [Basili et al. 1999] and determine the limits to which a study can generalise [Gómez et al. 2014].

As a start, the 208 studies mentioned in the theoretical part of the methodology are sourced from Hall et al. [2012]. Replicated papers found in the 208 studies will be selected. I will then perform reproducibility checks on those replicated papers that share their datasets: These checks set the path to answer RQ5 in chapter 5. Due to quality affecting defect prediction datasets [Gray et al. 2009], for each study that I will reproduce, data consistency checks will be performed. The aim is to identify studies that are reproducible and replicable, which can be presented as exemplar studies for researchers to emulate in achieving credibility (reproducible) and stability (replicable).

3.4 Investigations into factors affecting reproduction and model performance

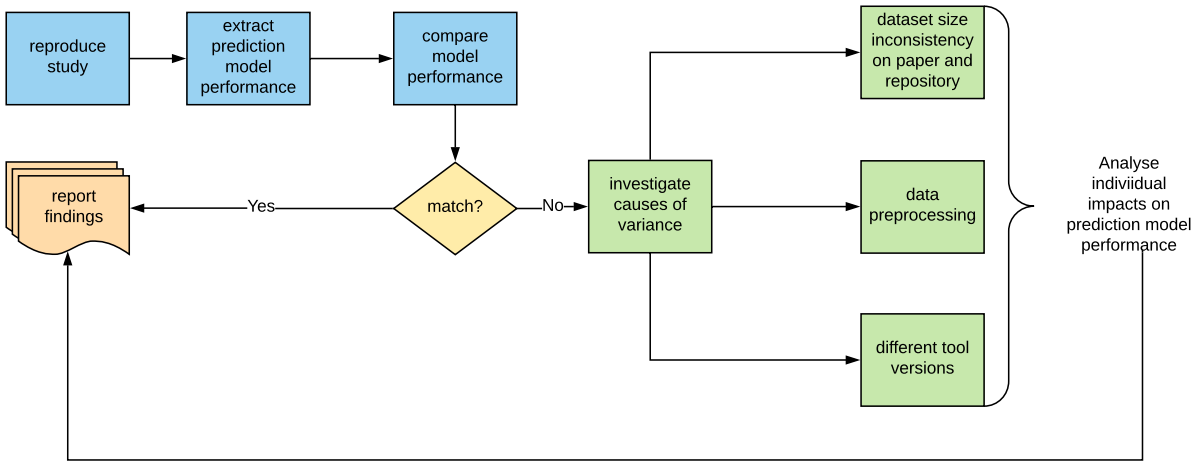


Figure 3.2: The process followed to investigate reproducibility failures and identify potential factors causing the failures

During the process of reproducibility checks I will extract and compare the prediction model performance for both the originals and reproduction studies. If the difference is higher than 5 percent, I will consider them different: a lower difference will indicate that a study is reproducible. I will then point out reproducible and replicable studies, and those that fail I will carry out further investigations on 3 main factors (data inconsistency, data preprocessing and tool variations) as to why they failed (Figure 3.2); this investigation completes the data needed to answer (RQ5 in chapter 6).

3.5 Summary

In this chapter I contribute a detailed methodology that aims to find replications, assess stability of findings between replications and original studies in order to identify exemplar studies that are reproducible and replicable.

In the next three chapters I provide the results achieved from the application of each aspect of my proposed methodology. The analyses and results of all three components of the methodology will be provided together with answers to five of my research questions. That is the analyses and results of:

- identifying and analysing replications studies,
- reproducing a systematically identified set of replicated studies, and
- investigating important factors that may affect the process of validating defect prediction experiments through reproduction based on their impact on prediction model performance.

4. Analysing the Replicability of Defect Prediction Studies

4.1 Introduction

The results in this chapter have been published Mahmood et al. [2018]. The paper is attached in section A.1. I was the lead-author of this journal paper. My role was to conduct the initial analysis, find the papers and write the first draft paper. The motivation for this chapter comes from one of the most significant threats discovered by Shepperd et al. [2014]'s meta-analysis mentioned in chapter 1; the research group that conducted the prediction study impacts model performance more than the main idea of the topic – the algorithms used for developing the prediction model [Madeyski and Kitchenham 2017]. The implication of this finding is that studies may not be reproducible and replicable by other researchers, which are important approaches used to demarcate science from nonscience [Gandrud 2013], establish matters of fact [Shapin et al. 1985] and to verify results so that the results may be trustworthy [Madeyski and Kitchenham 2017] and generalisable [Basili et al. 1999, Silva et al. 2014, Leek and Peng 2015].

The aim of this chapter is to answer RQ(1-4), in general, to determine the state of replication in defect prediction. I aim to contribute a systematically identified set of replication studies and analyse how consistent they are with the original studies they replicated. Knowing success and failure of replications provide information that these studies have the ability to be validated which may increase the confidence towards practitioners that these studies are reliable. Replicable and reproducible studies would serve as exemplar studies for other researchers to better understand replication and reproduction processes. I also aim to find characteristics of these replicated studies that are related to replication so that these may be

incorporated in new research studies to make them more accessible to replication.

The methodology I developed in chapter 3 (Figure 3.1) is applied in 6 stages: stage 1 uses the 208 studies from [Hall et al. 2012] as a base set to find any of the papers that has been replicated, stage 2 applies the inclusion criteria on the replication papers found (focuses on replications done by independent authors), stage 3 applies the data extraction, stage 4 categorises the identified replication papers into replication types, stage 5 identifies the factors (quality, citation counts, publication venue, impact factor, and data availability) associated with the frequency with which originals are replicated and stage 6 assesses the consistency of the findings between replications and original studies via agreements.

The following subsections provide results of the 6 stages of my proposed methodology (described in chapter 3, assesses research validity of a subject area) applied on the 208 studies from Hall et al. [2012]'s SLR.

4.1.1 Identification of papers

Table 4.1 shows the results of applying the methodology in chapter 3. Starting from the left the first column contains the original studies that were found to be replicated out of the 208 studies from the SLR. Only 13 out of the set of 208 original studies (13-ORS) were replicated by different researchers: 6% of the original studies, a much lower rate than the 94% non-replicated original studies; this means that the lack of replication is substantial. There are many studies that have not been confirmed to report valid results via replication.

The second column, replication studies, shows the 26 papers identified as having to replicate the 13-ORS. There are two replication studies by (Hamill and Goseva-Popstojanova, Hongyu Zhang) that replicate more than one original study, these two papers appear twice making the number of replication studies 26; these papers then appear twice on the list of the replication-references section. It is also important to

note that in the first column, three original studies (Andersson and Runeson, Lessmann *et al.*, Ostrand *et al.*) also conducted replications of other original studies making them appear as both original and replication studies (second column). It is because within them, certain components[1] of their study that are not part of the original study they replicated, is original and, also has been replicated by a different paper. For example, part of Lessmann et al. [2008][2] is a replication of Menzies et al. [2007] and the other aspect is a novel classifier benchmarking framework which Ghotra et al. [2015] replicated. Since part of Lessmann *et al.* is a replication and the other aspect is original, Lessmann *et al.* appears twice in Table 4.1 and on both lists of the replication and original-references sections.

4.1.2 Categorisation of replications to types

As explained in chapter 2 (Table 2.7) replication has different types (three in general; literal, operational, conceptual) and are based on the changes that can be made to the original's study components. The focus is on replications done by authors other than the original authors, operational replication, with 15 changes that could be made to the original study. 8 changes involve different authors and I tag these combinations (from A to H, see Table 2.7 in chapter 2). After identifying the 26 replication papers, I broke down each study into components that could be changed and mapped them to the respective type of replication. Table 4.1, third column, shows that all replication studies made many changes to the original study. Typically replications made three sets of changes to components of original studies.

[1] algorithms, datasets, experimental procedures; such as feature selection, data normalisation, cross-validation etc.

[2] Naïve Bayes classifier, datasets, cross-validation, the authors confirming that its a replication, not all studies mention this. Papers that do not mention replication but its synonym were returned by my search string and I read them in full to confirm that they are replications and then I successfully applied the Gómez et al. [2014]'s replication taxonomy to categorise studies into their respective replication types.

Table 4.1: 13 replicated original studies out of the 208 with their replication studies and the types of replications they performed

Original studies replicated	Replication studies	Operational rep.[1]
[D'Ambros et al. 2010]	[Mende 2010]	(A), (G)
[Andersson and Runeson 2007]	[Hamill and Goseva-Popstojanova 2015a]	(H)
	[Zhang 2008a]	(H)
[Lessmann et al. 2008]	[Ghotra et al. 2015]	(A), (H)
[Ostrand et al. 2005]	[Leszak 2005]	(H)
	[Mende and Koschke 2010]	(G)
[Fenton and Ohlsson 2000]	[Andersson and Runeson 2007]	(H)
	[Galinac Grbac et al. 2013]	(H)
	[Ostrand et al. 2005]	(H)
	[Zhang 2008b]	(H)
	[Devine et al. 2012]	(H)
	[Hamill and Goseva-Popstojanova 2015b]	(H)
[Menzies et al. 2007]	[Turhan and Bener 2007]	(G)
	[Zhang et al. 2007]	(G)
	[Lessmann et al. 2008]	(G)
	[Song et al. 2011]	(A),(H)
	[Singh and Verma 2014]	(H)
[Moser et al. 2008]	[Krishnan et al. 2013]	(H)
[Kim et al. 2007]	[Rahman et al. 2011]	(H)
[Zimmermann and Nagappan 2008]	[Tosun et al.]	(H)
	[Nguyen et al. 2010]	(H)
	[Premraj and Herzig 2011]	(H)
[Amasaki et al. 2003]	[Okutan and Yıldız 2014]	(H)
[Schröter et al. 2006]	[Duala-Ekoko and Robillard 2009]	(H)
[Zimmermann et al. 2007]	[Kpodjedo et al. 2011]	(H)
[Khoshgoftaar and Seliya 2002]	[Li et al. 2005]	(H)

[1]Replication name tags: (A) changed-experimenters, (G) changed-protocol/-operationalisation/-experimenters, (H) changed-protocol/-operationalisation/-populations/-experimenters. Tagging is an information retrieval concept that allows efficient retrieval of information by reducing unnecessary information [Hotho et al. 2006]. I tagged these replication names to easily allow identification of gaps between replication types performed.

Table 4.1 shows that many changes are made to studies:

1. Changed-experimenters (tag A, 3 papers)
2. Changed-protocol/-operationalisation/-experimenter (tag G, 5 papers)
3. Changed-protocol/-operationalisation/-populations/-experimenters (tag H, 21 papers)

Replications in which most components are changed together (tag H, 21 papers) dominates the other types of replications (A to G). With changed-experimenter (A) as the first change and (H) as the last, changes (B,...,F) have been omitted for all replications indicating gaps in steps that need to be taken during replications.

Table 4.1 shows that ([Mende 2010], [Song et al. 2011], [Ghotra et al. 2015]) replicate with sets of two study-component changes (A,G and A,H); the studies did multiple runs of a single original study. The first run reproduced the original study as it is, and the second run either modified the protocol (e.g. Mende [2010] changed D'Ambros et al. [2010]'s protocol by adding a cross validation step, Song et al. [2011] used feature selection that ensured the test instances are not seen by the prediction model), or change dataset (Ghotra et al. [2015] used less noisy data and new datasets, Song et al. [2011] also added more datasets). These multiple runs have implications for agreements between studies (i.e the need to account for agreements in all runs) and the need to categorise each run into its type of replication performed, based on each component change, though such multiple runs are generally good practice.

By breaking studies into components and sub-components I was able to synthesise data in the component structure from all studies (Table 4.3, Table 4.4, Table 4.5 for original studies, and Table A.1, Table A.2, Table A.3 in the appendix for replication studies). The data depicts a landscape of some of the sub-components; tools, algorithms, and statistical analyses used in defect prediction. The main components of a defect prediction study and sub-components of each including the number of times each sub-component is changed for all replications is given in Table 4.2. The Table 4.2 shows that the sta-

tistical test sub-component has the most changes compared to parameter tuning with the least changes. Replications tend to focus more on finding the most suitable statistical methods to describe data (e.g. [Zhang 2008a] suggests distribution of software faults are better described as Weibull distribution, not Pareto principle as originally proposed by [Fenton and Ohlsson 2000]). While tuning parameters of the prediction models to improve performance is considered the least (indicating that there may be many sub-optimal models being built).

Table 4.2: Study-components of Original Studies that were changed during Replication

Protocol	Stats	CrossVal	DataClean	Parameter tuning
	19	8	5	2
Operationalisation	IndepVar	DepVar	Algorithm	Tools
	15	4	12	14
Populations	Granularity	Domain	SourceCode	ProgLang
	14	12	11	7

Full field names from left to right and top to bottom: statistical analysis, cross validation, data cleaning, optimising parameters, independent and dependent variables, programming language

Table 4.3: Protocol: Original Studies

Org. Studies	Cross Val.	Parameter Tuning	Statistics	Data Cleaning
[Andersson and Runeson 2007]	No	No	Pearson product-moment correlation	Yes: Duplicate failures
[Lessmann et al. 2008]	Hold-out set	Yes	Friedmans test (rank classifiers), Nemenyi post hoc (statistical significance test on classifiers)	No
[Ostrand et al. 2005]	No	No	t-test	No
[Fenton and Ohlsson 2000]	No	No	Alberg diagrams	No
[Menzies et al. 2007]	10 by 10 randomised	No	Quartile chart	No
[Moser et al. 2008]	10 by 10 randomised	No	Kruskal-Wallis test	No
[Kim et al. 2007]	No	No	No	No
[Zimmermann and Nagappan 2008]	split-sample	No	Spearman correlation, Pearson, Nagelkerke (predictive power of logistic regression models), F-tests	No
[Amasaki et al. 2003]	No	No	Error rate, Fishers exact test (correlation between 2 variables)	No
[Schröter et al. 2006]	random splits	No	Two t-test, Spearman rank correlation	No
[D'Ambros et al. 2010]	50 by 10fold randomised	No	F-test (explanative significance), Spearman correlation (evaluating predictive power of models) with Spearman coefficient (skewed data)	No
[Zimmermann et al. 2007]	No	No	Spearman correlation, Pearson correlation	No
[Khoshgoftaar and Seliya 2002]	10 fold cross validation	No	Z-test	No

Table 4.4: Operationalisation: Original Studies

Org. Studies	Tools	Algorithms	Independent Var	Dependent Var
[Andersson and Runeson 2007]	No	No	Size (LOC)	Number of faults, fault density pre-release and post-release
[Lessmann et al. 2008]	YALE machine learning	Statistical(7), Nearest Neighbours(2), Neural Networks(3), SVMs(5), Tree-based(3), Ensembles (2)	static code metrics	Defective or Not-defective
[Ostrand et al. 2005]	VCS	Negative binomial regression	code age, programming language, logKloc,file status, release	Number of faults
[Fenton and Ohlsson 2000]	ERIMET (metrics),FCTOOL (formal description language)	No	Complexity (McCabe cyclomatic complexity), size (LOC), communication (SigFF;new and modified signals count, inter and intra modules)	Number of faults, fault density prerelease and postrelease
[Menzies et al. 2007]	WEKA	OneR, J48, and Naive Bayes	Static code metrics	Defective or Not defective
[Moser et al. 2008]	WEKA	logistic regression, Naive Bayes, J48 (version 8)	Process, change, static code	Defective or Not defective
[Kim et al. 2007]	Kenyon Infrastructure, APFEL (metrics), SVN, CVS	Least recently used (LRU)	spatial locality, temporal locality,Changed-entity and new-entity locality (churn)	Number of faults
[Zimmermann and Nagappan 2008]	MaX (dependency information tracker), Ucinet 6 (network metrics)	linear and logistic regression	Network measures on Dependency graphs, OO metrics, static code metrics	Number of defects, Defective or Not defective
[Amasaki et al. 2003]	Netica (bayesian belief network software)	Bayesian Belief Network	design (product size), effort (person-day), detected faults, test items	Number of residual faults at acceptance test
[Schröter et al. 2006]	R, BUGZILLA, CVS, SZZ	Linear and Ridge regression, Regression trees, SVM	import relationships (e.g. org.eclipse.ui) packages and imported classes	Number of defects, Defective or Not defective (postrelease)
[D'Ambros et al. 2010]	CVS, SVN, Bugzilla, Jira, Famix-Compliant OO model (scm metrics), Infusion (source code converter into FAMIX model), Moose (scm calculator), Churasco (history model, bug data extractor, classes linker, system files and bugs versioner)	linearReg	process, change , entropy of change , entropy, churn of source code, source code metrics, CK, OO	Number of defects (postrelease)
[Zimmermann et al. 2007]	Java parser (complexity metrics)	linear regression (ranking), logistic regression (classification)	static code metrics (complexity), structure of abstract syntax tree (no of nodes etc.)	Number of defects, Defective or Not defective (prerelease, postrelease)
[Khoshgoftaar and Seliya 2002]	S-Plus (advanced data analysis), EMERALD - Datrix (fault data collection)	least squares tree, s-plus, least absolute deviation	Design	Number of faults

Table 4.5: Populations: Original Studies

Org. Studies	Prog. Lang.	Domain	Granularity	Source	Availability
[Andersson and Runeson 2007],[Andersson and Runeson 2007]	C, Java	Telecom	Module	Commercial	Not shared
[Lessmann et al. 2008]	C, Java	Satellite, Flight, Storage	Module (predictions), code (metrics)	NASA, PROMISE	Shared
[Ostrand et al. 2005]	Java, C , Makefiles, sql, shell, html, other	Inventory System, Provisioning System	File (predictions)	Industrial	Not Shared
[Fenton and Ohlsson 2000]	No	Telecom	Module	Ericsson Telecom AB	Not shared
[Menzies et al. 2007]	C, Java	Satellite, Flight, Storage	Method	NASA, PROMISE	Shared
[Moser et al. 2008]	Java	IDE	File (predictions)	Eclipse 2.0, 2.1, 3.0	Not shared
[Kim et al. 2007]	C, C++, Java	Web server, Browser, Text editor, Version control, Database, IDE, Email client	File, method (predictions)	Apache 1.3, JEdit, Subversion, PostgreSQL, Columba, Eclipse, and Mozilla	Not shared
[Zimmermann and Nagappan 2008]	C++	Operating System	Binaries (predictions), Binaries (metrics)	Windows Server 2003	Not shared
[Amasaki et al. 2003]	No	Embedded software	development process (metrics), directed graphs	Industrial	Not shared
[Schröter et al. 2006]	Java	IDE	File, package (predictions), import packages and classes (metrics)	ECLIPSE plug-ins 52nos	Shared
[D'Ambros et al. 2010]	Java	IDE	Class (predictions, metrics)	Eclipse (Mylyn, Equinox, PDE, Lucene, Score)	Shared
[Zimmermann et al. 2007]	Java	IDE	files, packages (predictions, metrics)	Eclipse 2.0, 2.1, 3.0	Shared
[Khoshgoftaar and Seliya 2002]	Protel	Telecom	Modules (predictions), design documents (metrics)	Industrial	Not shared

4.1.3 Identification of factors associated with replications

As explained in chapter 2 and chapter 3, for all 208 original papers, I[3] selected contextual factors of studies a priori (based on the literature) at stage 5 of the methodology (quality, citation counts, publication venue, impact factor, and data availability) to determine their relation with replication. The extracted factors were analysed from each paper statistically[4]. A χ^2 test was used to establish the relationship between each binary factor (i.e. venues, citations etc.) and replications and Kendall's rank correlation [Kendall 1962] to test the relationship between citations and replications (as citations is continuous data, all factors are not assumed to come from a normal distribution of prediction studies).

Table 4.6: The 208 papers categorised as having four-phased quality of reporting and methodology ($quality_{4p}$), shared data, appeared in TSE w.r.t being replicated

	$quality_{4p}$		shared data		InTSE*	
Replicated	Yes	No	Yes	No	Yes	No
Yes	3	10	5	8	5	8
No	33	162	70	125	10	185

*chosen as TSE dominates in Table 4.9

Table 4.6 shows the number of original studies have been replicated or not based on their quality assessment (in Table 3.2), whether they share their data and whether they are published in TSE journal. The table shows few replicated studies pass $quality_{4p}$ and most of the replicated studies are appear in TSE.

Table 4.7 shows the data format of the papers with datasets. The table shows that there are few papers using formats other than arff. The small numbers do not allow a sound statistical analysis to be carried out for the affect of flexibility[5] on the ability to be replicated.

[3] I did this extraction independently and concurrently with David Bowes using SLuRp, discrepancies in the data were corrected and agreed upon. The statistical analysis was also validated by Bowes.

[4] using `R 3.3.1` open source statistical software.

[5] is one of the 8 factors González-Barahona and Robles [2012] suggested to affect reproducibility, the remaining factors overlap with Hall et al. [2012]'s $quality_{4p}$ on reporting sufficient information by a study to aid replication

Table 4.7: The number of papers which have available data and they were replicated

Data format (flexibility)	Not replicated	Replicated
arff	60	2
csv	0	1
csv, arff	2	1
csv, xml	1	0
excel	1	0
xml	3	1

There were 85 venues in which the 208 papers appeared (online-appendix[6]). Only 6 venues published papers that were subsequently replicated: PROMISE, MSR, ESEM, ISSRE, ICSE and TSE. TSE has the highest number of papers published with subsequent replications (Table 4.9). Table 4.8 shows that papers published in TSE are more likely to be replicated. We do not consider the impact factor of venues directly since, for non-replicated studies, impact factors are not available for many (63) publication venues.

Table 4.8: Statistical tests for assessing $quality_{4p}$, shared data, TSE and citations, individually against replications

Chi Square test	χ^2	p-value		
$quality_{4p}$ $*$ replication	0.322	0.570		
shared data $*$ replication	0.035	0.852		
InTSE $*$ replication	20.237	$<$ **0.0001**		
Kendall		z	τ	p-value
citations $*$ replication		4.7614	0.269	$<$ **0.0001**

Table 4.8 shows that a paper's influence (citations) has an impact on replication. However the quality of original papers or shared data used is not associated with subsequent replication.

Table 4.9 shows 10 of the 13-ORS have not passed the $quality_{4p}$ assessments. Replication not based on $quality_{4p}$ has ramifications on the validity of findings. For instance, referring back to Table 4.3, it shows data cleaning of the $quality_{4p}$ may have been overlooked or not reported, an indication

[6]https://bugcatcher.herts.ac.uk/replication/Online-Appendix.html

Table 4.9: The replicated original studies and their extracted contextual factors

Citations	Original Studies	Journal	ERA[1]	Impact[2]	$Quality_{4p}$ Assessment[3] failed at phase	Reps
128	[Andersson and Runeson 2007]	TSE	A*	4.423	phase1 No prediction done	2
577	[Lessmann et al. 2008]	TSE	A*	4.423	phase2 NASA Data used	1
535	[Ostrand et al. 2005]	TSE	A*	4.423	phase4 Model building	2
684	[Fenton and Ohlsson 2000]	TSE	A*	4.423	phase1 No prediction done	6
816	[Menzies et al. 2007]	TSE	A*	4.423	phase2 NASA Data used	5
361	[Moser et al. 2008]	ICSE	A	2.988	pass all	1
330	[Kim et al. 2007]	ICSE	A	2.988	phase4 Model building	1
393	[Zimmermann and Nagappan 2008]	ICSE	A	2.988	phase4 Model building	3
240	[D'Ambros et al. 2010]	MSR	C	1.876	pass all	1
43	[Amasaki et al. 2003]	ISSRE	A	1.383	phase2 Contextual information	1
159	[Schröter et al. 2006]	ESEM[4]	A	0.992	pass all	1
126	[Khoshgoftaar and Seliya 2002]	ESEM[5]	A	0.992	phase2 Contextual information	1
508	[Zimmermann et al. 2007]	PROMISE	U	0.001	phase2 Contextual information	1

[1] CORE contributed to ERA rankings. Both rankings agree except on ICSE; A by ERA, A* by CORE (TSE not ranked)

[2] Source is journalmetrics, 2015 source normalised impact (SNIPA); takes average citation per paper of a journal in subject area

[3] QA details summarised in Table 3.2. [4]ISESE, [5]METRICS are now part of ESEM http://www.esem-conferences.org/history.php

that some findings may be erroneous. It is particularly true for the noisy NASA datasets used by 59 original studies (Table OA.1 online-appendix).

Table 4.10: Descriptions of replicated original studies based on the type of data source and defect data sharing

Source Code	Shared data	Org. papers	Rep. papers
Closed	No	6	15
	Yes	2	6
Open	No	2	2
	Yes	3	3

Table 4.10 shows that 21 of 26 replication studies replicated original studies which were based on closed source industrial data (these will have needed to be replicated with different datasets). This suggests that studies based on closed source industrial data may be more attractive for replication.

4.1.4 Assessment of agreements between studies

Assessing agreements is difficult when dealing with replication studies. Silva et al. [2014] reported that they could not asssess agreements in their review of replication studies because of reporting inconsistencies of replications. In this section I show how I assess agreements. My approach is based on the synthesis I did on how Andersson and Runeson [2007] reported their replication of Fenton and Ohlsson [2000]. The Fenton and Ohlsson [2000] study outlined clearly whether a statistical claim tested was (none, weak, strong)[7]. This measuring style is important because the margin of error of the dependent variable (for example, the number of defects found by a model measured using MCC) may be much wider than that of the original study. Replication has many differences in terms of data, tools, and method, so, the inherent differences are likely to cause differences in the results. Therefore, I focus on assessing the outcome of the hypotheses tested. The hypotheses outcome is a claim that a relationship between two entities in the original study and its replication or otherwise. If the outcome is not clear I report what the authors report. If none are available I conclude that the agreement is unknown.

[7] An alternative can be found in Psychology by Patil et al. [2016] - the study suggested the use of confidence intervals.

Table 4.11: 13 replicated original studies out of the 208 with their replication studies, replication types and agreements between studies (this table is not the same as Table 4.1). This Table contains the agreements between originals and their replications showing that most replications agree based on the reported hypotheses.

Replicated original studies	Replication studies	Agreements	Operational rep.[1]
[D'Ambros et al. 2010]	[Mende 2010]	Yes, Yes	(A), (G)
[Andersson and Runeson 2007]	[Hamill and Goseva-Popstojanova 2015a]	Yes	(H)
	[Zhang 2008a]	No	(H)
[Lessmann et al. 2008]	[Ghotra et al. 2015]	Yes, No	(A), (H)
[Ostrand et al. 2005]	[Leszak 2005]	Partial	(H)
	[Mende and Koschke 2010]	Unknown	(G)
[Fenton and Ohlsson 2000]	[Andersson and Runeson 2007]	Partial	(H)
	[Galinac Grbac et al. 2013]	Partial	(H)
	[Ostrand et al. 2005]	Yes	(H)
	[Zhang 2008b]	No	(H)
	[Devine et al. 2012]	Yes	(H)
	[Hamill and Goseva-Popstojanova 2015b]	No	(H)
[Menzies et al. 2007]	[Turhan and Bener 2007]	Yes	(G)
	[Zhang et al. 2007]	Yes	(G)
	[Lessmann et al. 2008]	Yes	(G)
	[Song et al. 2011]	Yes, No	(A),(H)
	[Singh and Verma 2014]	Yes	(H)
[Moser et al. 2008]	[Krishnan et al. 2013]	Yes	(H)
[Kim et al. 2007]	[Rahman et al. 2011]	Yes	(H)
[Zimmermann and Nagappan 2008]	[Tosun et al.]	Yes	(H)
	[Nguyen et al. 2010]	Yes	(H)
	[Premraj and Herzig 2011]	Yes	(H)
[Amasaki et al. 2003]	[Okutan and Yıldız 2014]	Unknown	(H)
[Schröter et al. 2006]	[Duala-Ekoko and Robillard 2009]	Yes	(H)
[Zimmermann et al. 2007]	[Kpodjedo et al. 2011]	Unknown	(H)
[Khoshgoftaar and Seliya 2002]	[Li et al. 2005]	Yes	(H)

Agree:	replications that confirm original findings	18
Disagree:	replications that do not confirm original findings	5
Partial:	replications that confirm part of the original findings	3
Unknown:	replications that do not report any of the above	3

In my work I propose and use 3 criteria for measuring agreements as follows (the summarised results are in Table 4.11, see Table A.4 in the appendix for the full data):

1 If a study performed a reproduction study (also tagged A in Table 4.11), model performance values should be compared between the original and its reproduction study. If the percentage difference is greater than 5%, then it is a disagreement.

2 If a study performed a replication (also tagged G and H above among other types), the outcome of the same hypotheses tested in both original and its replication should be compared. That is, what both studies reported, for example statistical power being strong, weak, absent (see example in Fenton and Ohlsson [2000] replicated by Andersson and Runeson [2007]).

3 If (1) and (2) are not possible, the agreement reported by the replication confirming the original's findings should be used otherwise it is unknown.

Table 4.11 shows the number of replication studies that agree or disagree with the conclusions of original studies. Table 4.11 shows that 17 experiments[8] agreed with the conclusions stated in the original accompanying paper, which indicates that 61% of the experiments are successful replications. There are 5 experiments (18%) that disagree and 3 experiments (11%) assessed as partial agreement. As I mentioned earlier agreements are difficult to extract. I now give worked examples based on my agreement criteria.

4.1.4.1 Worked examples for synthesising agreements within original and replication studies

Under this section the agreement criteria is applied. The application will show how to determine agreements in three ways. The results of each criterion are illustrated from the original and replication papers.

Example based on agreement criteria 1 - Comparing model performance values:

Song et al. [2011] reproduced Menzies et al. [2007] and then replicated the study. In the first run the replication agrees with Menzies et al. [2007]. Prediction values on the original and replication are very close. For example, on datasets CM1, KC3 and KC4 the values in the order (original, replication) are (69.5, 69.5), (69.7, 70.8), and (68.1, 69.1) respectively.

Example based on agreement criteria 2 - Comparing outcome of the same hypothesis tests:

An example of partial agreement is the replication by Andersson and Runeson [2007] of the Fenton and Ohlsson [2000] study. The replication reported confirmation of 3 out of 5 hypotheses tested, thus I categorised this as partial agreement. An example of one of the hypothesis is "High fault incidence in Functional Test implies the same in System Test". Fenton and Ohlsson [2000] found No support for the

[8] some papers conduct more than one experiment, there are 26 papers running 29 experiments

hypothesis and its replication by Andersson and Runeson [2007] found Strong support.

Example based on agreement criteria 3 - Clear reporting of the agreement:

Ghotra et al. [2015] replicated Lessmann et al. [2008] in 2 runs and reported an agreement as *"First, we apply the replicated procedure to the same (known-to-be noisy) NASA dataset, where we derive similar results to the prior study, i.e., the impact that classification techniques have appear to be minimal. Next, we apply the replicated procedure to two new datasets: (a) the cleaned version of the NASA dataset and (b) the PROMISE dataset, which contains open source software developed in a variety of settings (e.g., Apache, GNU). The results in these new datasets show a clear, statistically distinct separation of groups of techniques."* [Ghotra et al. 2015].

Performing multiple runs of replication affects agreement outcomes. Song et al. [2011] did two runs (1st run fell under agreement criteria 1, because it is a reproduction study). The second run, Song et al. [2011] disagreed and reported a flaw in Menzies et al. [2007]'s attribute selection approach. The flaw is that the test data included information seen by the prediction model which inflated its performance. Ghotra et al. [2015] did 2 replication runs of [Lessmann et al. 2008]. The first run was based on uncleaned NASA data (including duplicate and inconsistent instances, see [Petrić et al. 2016]) to confirm if no single classifier is best as in the original study [Lessmann et al. 2008]. The Friedman test used in Lessmann et al. [2008] showed the ranking of model performances are not random; subsequently Nemenyi post hoc test was applied to detect which of the classifiers differed significantly. Ghotra et al. [2015] agree with Lessmann et al. [2008] in the first run with the same data and different statistics, but disagrees in the second run with a cleaned dataset curated by Shepperd et al. [2013] and different statistics. In the second run, Ghotra et al. [2015] reported;

"*We used the Scott-Knott test to overcome the confounding issue of overlapping groups that are produced by several other post hoc tests, such as Nemenyis test [13], which was used by the original study.*

Nemenyis test produces overlapping groups of classification techniques, implying that there exists no statistically significant difference among the defect prediction models trained using many different classification techniques"[Ghotra et al. 2015]. The curated data by Shepperd et al. [2013] has been cleaned further by Petrić et al. [2016]. The data errors found during this further cleaning may have also affected previous models.

4.2 Conclusion

In this chapter, my proposed methodology for assessing research validity of a subject area through replication provide new evidence (and contributes answers to RQ(1-4)) that only 13 (6%) of the 208 studies are replicated, 10 of which fail a quality check. Replication seems related to original papers appearing in the Transactions of Software Engineering (TSE) journal. The number of citations an original paper had was also an indicator of replications. In addition, studies conducted using industry closed source data seem to have more replications than those based on open source data. These findings show that very few defect prediction studies are replicated. The lack of replication means that it remains unclear how reliable defect prediction is. Where a paper has been replicated, 11 (38%) out of 29 replications revealed different results to the original study. In the next chapter 5 I perform reproducibility checks on the replicated studies having dataset in order to increase the validity of my findings; this is also motivated by how reproducibility experiments are measured with the expectation of an error margin between an original and its reproduction not being more than 5%.

5. Analysing the Reproducibility of Defect Prediction Studies

5.1 Introduction

The aim of this chapter is to answer (RQ5), that is to assess whether replicated defect prediction studies are reproducible. Secondly, to contribute factors that may aid or hinder reproduction. Studies that are reproducible and replicable would serve as exemplar studies with useful characteristics. Researchers could incorporate such characteristics to increase the ability of their work to be validated. Studies with credible and stable results are valuable to their community and scientific fields [Ioannidis 2017].

Of the 13-ORS I identified in chapter 4 only 5 have their data available. I reproduce the 5 studies using the Java based data mining library WEKA and R statistical software. Java and 'R' were used to write 4 of the 5 original experiments I reproduce. Only one study used Java and Matlab. I am more conversant with Java and R, and for consistency and ease of use, I did not use Matlab. I present the reproduction experiments in the following format:

- the purpose of the original study,
- the dataset used,
- the classifiers used,
- the tools used where possible,
- the method (i.e. experimental procedure of the original),
- my reproduction procedure with changes I make and assumptions made about the original, and
- a comparison of my results and the original study's together with any missing information.

5.2 Reproducing baseline results of static code metrics on the original NASA datasets

The aim of Menzies et al. [2007] was to provide baseline results on NASA datasets and to provide evidence that static code metrics are useful for predicting defects. As such these results need to be as reliable as possible.

Data: The Menzies et al. [2007] study used eight NASA datasets[1]. Each dataset is created from four systems (Spacecraft, Storage, Database and Flight). The metrics from McCabe [1976]; e.g. cyclomatic complexity *v(g)* shows pathways in modules using flow graph of data points and connecting arcs. The data points are program statements. The connecting arcs show movement of control from one statement to another. The Halstead [1977] attributes measure the difficulty of reading code. For example, Halstead counts the number of operators, and operands etc., as metrics.

Classifier: Naïve Bayes was used as a classifier for predicting defects. The classifier estimates the probability that a new module is either defective or not. The estimation is based on previously calculated probabilities of defective or non-defective dataset instances. The probabilities of the defective and non-defective classes are calculated independently (e.g. *v(g)* only, or LOC). The classifier assumes each feature contributes individually to the class probability. A summary of classifiers is found in Figure 5.2.

Tool: The original study used WEKA data mining tool kit [Witten and Frank 2005] for their experiments.

Experimental procedure of Menzies et al. [2007]: The datasets were *preprocessed* using a logarithmic filter. Natural log (*ln*) of all numeric values were taken. The numerical $values < 0.000001$ were replaced with *ln* (0.000001) to prevent infinity errors with *ln* (0). The log transform is an important step to even out the distribution of numeric values of features within datasets. Many small and large values (0.003,

[1] now moved to https://zenodo.org/communities/seacraft Accessed: 12th Nov 2017

726) are scaled to improve prediction performance [Menzies et al. 2007]. *Feature selection* was done using Information Gain (InfoGain). It is an aspect of information theory that uses a Ranker to rank features in the defect dataset. The models were built and evaluated using ($m = 10 \times n = 10$) fold cross validation. The evaluation prevents ordering problems and increases reliability of results [Menzies et al. 2007]. The (m) is the number of times the experiment is run and the dataset is randomised before being divided into (n) folds. 9 folds for training and 1 fold for testing. The performances were evaluated using probability of detection and false alarm (known as recall and false alarm rate), see chapter 2. It is important to note that these two performance measures do not provide a holistic performance of the model. The two performance measures are not based on the whole confusion matrix unlike Mathews Correlation Coefficient (MCC, see chapter 2).

Table 5.1: Model performance measure from Menzies et al. [2007] set of replications reporting different performance measures. Our reproduction with values in parentheses showing % differences $((rep - org)/org) * 100$. Positive values indicate that replication is higher.

	Naive Bayes						
	recall					auc	balance
Data	Menzies *et al.*	Mahmood (% diff)	Turhan & Bener	Zhang *et al.*	Singh & Verma (% diff)	Lessmann *et al.*	Song *et al.*
pc4	98	87 (**-11**)	–	–	72.6 (2)	85	83
pc3	80	79 (-1)	–	–	80.6 (**14**)	81	71
kc4	79	80 (1)	–	–	–	68	71.9
kc3	69	78 (**13**)	–	–	99 (**39**)	83	74
pc1	48	73 (**52**)	–	–	66.2 (**-7**)	79	65
cm1	71	77 (**8**)	–	–	81.5 (**15**)	72	73
pc2	72	86 (**19**)	–	–	83.3 (**17**)	85	82
mw1	52	78 (**50**)	–	–	100 (**41**)	80	71
avg	71	80 (**12**)	64 (**-10**)	85 (**20**)	83 (**17**)	79	74

Reproducing Menzies et al. [2007] and the changes made: I performed the Menzies et al. [2007] experiment[2] using the 'same' data, tool and procedure. I ran 1,000 iterations of 10-fold cross validation to build the prediction model. These several runs reorder the instances in each run with a different *seed* and the results for each dataset is an average of the *recall*; this improves reliability and minimises variance in the results [Kohavi et al. 1995].

[2]I ran the experiment with the error in the study that Song et al. [2011] pointed out (feature selection was performed on both train and test sets – nullifies prediction). This is strictly for the purpose of being able to reproduce a study and to identify factors aiding or hindering the process.

Comparing the results: In chapter 4 I found that Menzies et al. [2007] had been replicated in five studies. Since these five studies use the same data, I extracted the results of these five papers. I included my reproduction results and compared components (data, classifiers, feature selection, preprocessing, see Table 5.2 for details) across all of the studies. The aim is to assess internal validity of studies with minimal changes to the original study. Hence, I extracted components that are as close to the original as possible. In Table 5.1 results for [Turhan and Bener 2007], [Zhang et al. 2007], and Singh and Verma [2014] are > 5%. An average recall performance of (64 and 85) for the replications, disagreeing with [Menzies et al. 2007] (71). The results are not reproducible.

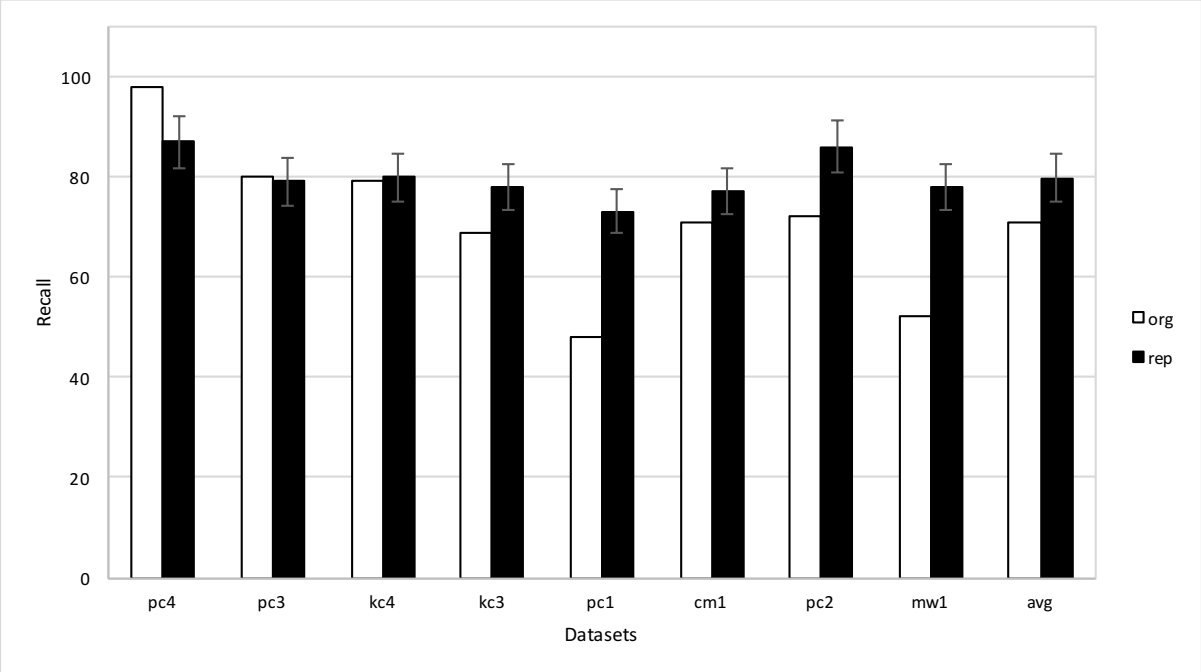


Figure 5.1: The bar plot of Menzies *et al.* and my reproduction results showing an experiment using Naïve Bayes classifier on 8 NASA datasets. The model performance is measured using recall (actual number of defectives found from total defectives) and only two datasets results on pc3 and kc4 are reproducible. The 9th data point is the average over all datasets.

In the case of Lessmann et al. [2008], a different measure (auc) was reported. Song et al. [2011] reported one performance measure (balance). These results show that reporting inconsistencies between replica-

tions and original studies make it difficult to confirm agreements and check reproducibility of replicated studies.

In Table 5.1, it is noteworthy that the differences in my reproduction (same data, tools, and method) results are as wide as the replications (different permutations of data, tools, and method). Even though these replication studies report that their results confirm those of the original studies. The results of my reproduction study in Table 5.1 and Figure 5.1 disagree with the original except for two datasets (pc3, kc4). In addition, some of the features from the output of the feature selection did not match what was reported in the original paper for datasets (kc3, pc1, pc2, mw1). The input and output of the feature selection process were reported in the original paper and so I was able to compare with my result. The results in each case are mixed despite matching closely all study components. Doing data consistency checks is important. I compared the proportion of defective instances and total number of instances of the datasets reported in the original paper and the downloaded version (Table 5.2). These dataset variations also cut across all Menzies *et al.* set of replication studies.

Table 5.2: Data consistency check showing variations across all Menzies *et al.* replications

Authors	Datasets	Menzies et al.	Turhan, Bener	Lessman et al.	Zhang et al	Song et al.	Singh, Verma	Mahmood
\% defectives	pc4	12	12	13.21	12.21	12.21	12	12.22
	pc3	10	10	10.59	10.23	10.24	10	10.5
	kc4	49	49	48.8	-	48.8	-	48.8
	kc3	9	9	9.39	9.39	9.39	9	9.39
	pc1	6	6	7.18	6.86	6.87	6.94	6.87
	cm1	9	9	9.5	9.5	9.5	9.83	9.5
	pc2	0.4	0.6	0.51	0.41	0.41	6	0.41
	mw1	7	7	7.69	7.69	7.69	7	7.69
Data Balance	pc4	24	24	26.42	24.42	24.42	24	24.44
	pc3	20	20	21.18	20.46	20.48	20	21
	kc4	98	98	97.6	-	97.6		97.6
	kc3	18	18	18.78	18.78	18.78	18	18.78
	pc1	12	12	14.36	13.72	13.74	13.88	13.73
	cm1	18	18	19	19	19	19.66	19
	pc2	0.8	1.2	1.02	0.82	0.82	12	0.82
	mw1	14	14	15.38	15.38	15.38	14	15.38
Instances	pc4	1458	1458	1347	1458	1458	1458	1458
	pc3	1564	1563	1511	1563	1563	1563	1563
	kc4	126	125	125	-	125	-	125
	kc3	459	458	458	458	458	459	458
	pc1	1108	1107	1059	1107	1107	1107	1107
	cm1	506	505	505	505	505	498	505
	pc2	5590	5589	4505	5589	5589	5590	5589
	mw1	404	403	403	403	403	403	403
Features Selected	pc4	3	not reported	37	3	not reported	not reported	3
	pc3	3	not reported	37	3	not reported	not reported	3
	kc4	3	not reported	13	3	not reported	not reported	3
	kc3	3	not reported	39	3	not reported	not reported	3
	pc1	3	not reported	37	3	not reported	not reported	3
	cm1	3	not reported	37	3	not reported	not reported	3
	pc2	2	not reported	37	3	not reported	not reported	3
	mw1	3	not reported	37	3	not reported	not reported	3
Feature Names Selected	kc3	loc_executable, L, T	not reported	in paper	LOC,V(g),V	in paper	not reported	loc_executable, loc_total, number_of_lines
	pc1	call_pairs, num_uniqe_operands, number_of_lines	not reported	in paper	LOC,V(g),V	in paper	not reported	loc_blank, num_uniqe_operands, number_of_lines
	pc2	loc_comments, percent_comments	not reported	in paper	LOC,V(g),V	in paper	not reported	loc_comments, halstead_error_est, percent_comments
	mw1	halstead_error_est, node_count, num_unique_operands	not reported	in paper	LOC,V(g),V	in paper	not reported	halstead_error_est, halstead_length, num_unique_operands
Granularity	-	method	method	method	component	method	method	method
Classifier	pc4	NB	NB	NB	NB	NB	CBC	NB
Feature Selection	-	infoGain	infoGain	tuning	manually	infoGain	none	infoGain
Preprocess	-	logFilter	logFilter	infeasible	datacleaning	logFilter	discretization	logFilter

5.3 Reproducing a framework for comparing defect prediction classification models

The aim of Lessmann et al. [2008] was to find the best of 22 classifiers over 10 NASA datasets. The benchmark results show no single classifier dominates.

Data: The study used 10 NASA datasets from the same source as the experiment in section 5.2.

Classifier: The study compared 22 classifiers from six classifier families as follows (summarised in Figure 5.2):

1. Statistical classifiers
2. Nearest neighbour methods
3. Neural Networks
4. Support vector machine-based classifiers
5. Decision tree approaches
6. Ensemble methods

Tool: Rapid Miner is a Java based data mining tool (formerly known as YALE Workbench) and Matlab development environment.

Experimental Procedure of [Lessmann et al. 2008]: The experiment is summarised in the pseudo code in Figure 5.3. Each dataset is randomised and then split into 2/3 for train set and 1/3 for test set, a procedure known as percentage split. The 22 classifiers are grouped into three based on the type of training procedure. The first group of classifiers do not require tuning, and are trained directly on 2/3 train set. The second group of classifiers require their parameters to be tuned (SVM, k-NN etc.) using a pool of parameter values to tune with cross-validation. The 2/3 train set is split in (n) folds, ($n-1$) is the sub train set and the last fold is the sub test set.

Classification model		Philosophy
Statistical classifiers		Strive to construct a Bayes optimal classifier by estimating either posterior probabilities directly (LogReg), or class-conditional probabilities (LDA, QDA, NB, BayesNet) which are subsequently converted into posterior probabilities using Bayes' theorem. LDA/QDA assume a multivariate Gaussian density function, whereas NB is based on the assumption that attributes are conditionally independent, so that class-conditional probabilities can be estimated individually per attribute. BayesNet extends NB by explicitly modeling statements about independence and correlation among attributes. LARS adopts a different approach and consists of a multivariate linear regression model and heuristics to shrink the number of features. RVM has been proposed as an extension of the SVM (see below) which avoids the need to tune certain hyperparameters and may incorporate kernel functions SVMs are unable to process.
Linear Discriminant Analysis[2, 3]	(LDA)	
Quadratic Discriminant Analysis[2, 3]	(QDA)	
Logistic Regression[2, 3]	(LogReg)	
Naïve Bayes[1]	(NB)	
Bayesian Networks[1]	(BayesNet)	
Least-Angle Regression[2]	(LARS)	
Relevance Vector Machine[2] [62]	(RVM)	
Nearest neighbor methods		Belong to the group of analogy-based methods which classify a module by considering the *k* most similar examples. The definition of similarity differs among algorithms. An Euclidian distance is used in *k*-NN whereas K* employs an entropy-based distance function.
k-Nearest Neighbor[1]	(k-NN)	
K-Star[1] [11]	(K*)	
Neural Networks		Mathematical representations inspired by the functioning of the human brain. They depict a network structure which defines a concatenation of weighting, aggregation and thresholding functions that are applied to a software module's attributes to obtain an approximation of its posterior probability of being fp. The study includes two types of MLP classifiers which incorporate different approaches to avoid overfitting the training data, i.e. weight decay and Bayesian Learning.
Multi-Layer Perceptron[2, 4]	(MLP)	
Radial Basis Function Network[1]	(RBF net)	
Support vector machine-based classifiers		Utilize mathematical programming to optimize a linear decision function that discriminates between fp and nfp modules. A kernel function enables more complex decision boundaries by means of an implicit, nonlinear transformation of attribute values. This kernel function is polynomial for the VP classifier, whereas SVM and LS-SVM consider a radial basis function. L-SVM and LP are linear classifiers.
Support Vector Machine[2]	(SVM)	
Lagrangian SVM[2] [40]	(L-SVM)	
Least Squares SVM[2] [61]	(LS-SVM)	
Linear Programming[2]	(LP)	
Voted Perceptron[1] [22]	(VP)	
Decision tree approaches		Recursively partition the training data by means of attribute splits. The algorithms differ mainly in the splitting criterion which determines the attribute used in a given iteration to separate the data. C4.5 induces decision trees based on the information-theoretical concept of entropy, whereas CART uses the Gini criterion. ADT distinguishes between alternating splitter and prediction nodes. A prediction is computed as the sum over all prediction nodes an instance visits while traversing the tree.
C 4.5 Decision Tree[1]	(C 4.5)	
Classification and Regression Tree[2]	(CART)	
Alternating Decision Tree[1] [21]	(ADT)	
Ensemble methods		Meta-learning schemes that embody several base-classifiers. These are built independently and participate in a voting procedure to obtain a final class prediction. RndFor incorporates CART as base learner, whereas LMT utilizes LogReg. Each base learner is derived from a limited number of attributes. These are selected at random within the RndFor procedure, whereby the user has to predefine their number. LMT considers only univariate regression models, i.e. uses one attribute per iteration, which is selected automatically.
Random Forest[1] [7]	(RndFor)	
Logistic Model Tree[1] [36]	(LMT)	

[1] Classifier is implemented using the YALE workbench [45].
[2] Classifier is implemented using the MATLAB environment.
[3] These classifiers fail to produce a classification model if all attributes are used. Therefore, they are trained in conjunction with a backward-feature elimination heuristic [25] (see also Appendix I).
[4] Subsequently, we use the abbreviation MLP-1 to refer to a multi-layer perceptron neural network which has been trained with a weight decay penalty to prevent overfitting, whereas MLP-2 represents a network which uses a Bayesian learning paradigm (see also Appendix I).

Figure 5.2: The classifiers and their descriptions from Lessmann *et al.* showing six groups of classifier families each differentiated by certain characteristics.

```
Input: D=List of datasets C=List of classifiers
  P=Dictionary of hyperparameter settings per classifier
Output: auc
  Initialise with:  noisy data
  LOOP Process
  for each d in D do
    train = randomly select 2/3 of d
    test = d - train
    for each c in C do
      p_opt = ModelSel (train, c, P[c] )
      model = BuildClassifier (train, c, p_opt )
      auc[c, d] = ApplyClassifier (test, model)
    end for
  end for
  #---------------------------------------
  ModelSel (data, classifier, hyperparameters)
  for i = 1 to 10 do
    crossval = generate 10 bins from data
    validate = crossval[i]
    learn = crossval - validate
    for each p in hyperparameters do
      model = BuildClassifier (learn, classifier, p)
      cv_auc[p, i] = ApplyClassifier (validate, model)
      auc = compute mean performance over cross-validation bins
    end for
  end for
  return  hyperparameters[max(auc)]
  BuildClassifier(data, classifier, para)
  # Train classifier on data with hyperparameters = para
  ApplyClassifier(data, model)
  # Compute AUC of model on data
```

Figure 5.3: Experimental Scheme of Lessmann et al. [2008]

A 10 by 10 fold cross-validation was used. The third set performs better when correlated features in the defect datasets are removed using feature selection. The full list of classifiers and parameters are in Table 5.3. The best model evaluated using Area Under ROC (AUC) is selected from any of the procedures. The selected model is then re-trained on the whole training set. The trained model is finally used to make predictions on the 1/3 test set.

Reproducing Lessmann et al. [2008] and the changes made: I attempted to use the tools as in the original study. However, I was more conversant with Weka [Hall et al. 2009] unlike the tools used by the original. Tools may also cause variation in the results and could invalidate a reproduction study. Lincke et al. [2008] compared metric tools on the same system and found variations in the software metrics calculated. The study concluded that some results are tool dependent. In this case, a tool comparison (Rapid miner, Matlab and Weka) was not done and is beyond the scope of this work. Though it would

Table 5.3: The names and classifier parameters taken from Lessmann et al. [2008] study arranged by original tools and the training and testing procedures

Tool: RapidMiner	Parameters	Procedure	Tool: Matlab	Parameters	Procedure
NB	-	NoModSel (2/3HoldOut)	LDA	BackFeatElim	FeatSel as ModSel
k-Star	-	NoModSel (2/3HoldOut)	QDA	BackFeatElim	FeatSel as ModSel
BayesNet	searchTechs (K2, sim annealing, tabuSearch, hill climbing, tree-Aug Na ive Bayes)	ModSel as 10CV	LogReg	BackFeatElim	FeatSel as ModSel
kNN	nearNeighbours (1,3,5, . . . ,15)	ModSel as 10CV	LARS (not found)	-	NoModSel(2/3HoldOut)
RBFNet	clusterCentres (1to10)	ModSel as 10CV	RVM (not found)	-	NoModSel(2/3HoldOut)
VotedPercep	polyKernal (1 to 6)	ModSel as 10CV	Linear Prog. - LP (not found)	-	NoModSel (2/3 HoldOut)
DecTree	prun + LapSmoo + SubtreeRaise (0.05,0.1,. . . ,0.7) prun (0.05,0.1,. . . ,0.7) unprun + LapSmoo + SubtreeRaise (0.05,0.1,. . . ,0.7) unprun (0.05,0.1,. . . ,0.7)	ModSel as 10CV	MLP-1	hidden (4,5,. . . ,28), decay(0.1,0.2)	ModSel as 10CV
AlterDecTree (ADTree)	boostinIterations (10 to 50)	ModSel as 10CV	MLP-2 (not found)	bayesian learning	ModSel as 10CV
LogisModTree (LMT)	defaultPrun (100 iterations overall)	ModSel as 10CV	SVM	rbf kernel, grid-search (c,gamma)	ModSel as 10CV
RndFor	noOfTrees(10,50,100,250,500,1000) noOfRandomselAttributes([0.5;1,2]dot, sqrtM) M is no of attributes in data	ModSel as 10CV	Lagrangian-SVM (not found)	regulParam (log(C)[-6,-5,. . . ,20]	ModSel as 10CV
			LeastSquares SVM (not found)	rbf kernel, grid-search (c,gamma)	ModSel as 10CV
			ClassRegTree-CART	prun + LapSmoo + SubtreeRaise (0.05,0.1,. . . ,0.7) prun (0.05,0.1,. . . ,0.7) unprun + LapSmoo + SubtreeRaise (0.05,0.1,. . . ,0.7) unprun (0.05,0.1,. . . ,0.7)	ModSel as 10CV
Model Selection	Tuning by 10 by 10 fold Cross-Validation or Feature Selection				
No Model Selection	2/3 HoldOut Only				

be valuable to conduct such an analysis when there are inexplicable differences in the outcome of the reproduction studies. 12 classifiers available in the Weka tool kit were trained and tested on the 10 NASA datasets. 1,000 runs of the procedure were done. In each run the datasets were randomised to handle ordering effects on classifier performance [Menzies et al. 2007]. The remaining procedure and parameters were applied as described in Figure 5.3.

Comparing the results: Table 5.4 and Figure 5.4 show that the classifiers with the most differences compared to the original Lessmann et al. [2008] study are SVM on all datasets and VP on 6 datasets. Overall, half of the results (60 model performances) are reproducible. Despite using the exact parameters reported by the original study in Table 5.3, it is not clear why the remaining half of the results failed to be reproduced. There may be unreported processes done with these classifiers that could be missing in the paper in addition to the variations in the datasets used (Table 5.2). I focus on the SVM and VP because the two classifiers failed the most. Further investigation to find possible explanations for this failure are reported in chapter 6.

Table 5.4: Lessmann et al. [2008] and my reproduction results showing both results, original as org and reproduction as rep. The negative percentage difference indicates that the my model's performances are lower and higher for the positive differences. Half of the results are reproducible, 60 model performances are 5% and below, the remaining are above 5%. SVM and VP are outliers with the most differences above 5% across most of the datasets.

Datasets	CM1			KC1			KC3			KC4			MW1			JM1			PC1			PC2			PC3			PC4		
classifier	org	rep	diff	org	rep	diff	org	rep	diff	org	rep	diff	org	rep	diff	org	rep	diff	org	rep	diff	org	rep	diff	org	rep	diff	org	rep	diff
BayesNet	79	74	**(-6)**	75	79	(5)	83	81	(-2)	80	81	(1)	82	75	**(-9)**	73	71	(-3)	84	81	(-4)	85	83	(-2)	80	80	(0)	90	89	(-1)
NaiveBayes	72	76	**(6)**	76	79	(4)	83	82	(-1)	68	77	**(13)**	80	77	(-4)	69	69	(0)	79	76	(-4)	85	87	(2)	81	77	(-5)	85	84	(-1)
LibSVM	70	51	**(-27)**	76	51	**(-33)**	86	50	**(-42)**	77	60	**(-22)**	65	50	**(-23)**	72	53	**(-26)**	80	56	**(-30)**	85	50	**(-41)**	77	53	**(-31)**	92	50	**(-46)**
MultilPer	76	72	(-5)	77	79	(3)	79	75	(-5)	80	76	(-5)	77	73	(-5)	73	71	(-3)	89	74	**(-17)**	93	83	**(-11)**	78	78	(0)	95	90	(-5)
RBFNetwork	58	72	**(24)**	76	78	(3)	68	77	**(13)**	73	75	(3)	65	72	**(11)**	69	69	(0)	64	72	**(13)**	79	76	(-4)	78	75	(-4)	79	82	(4)
VotedPerc	72	54	**(-25)**	76	77	(1)	74	62	**(-16)**	73	75	(3)	73	52	**(-29)**	54	54	(0)	75	53	**(-29)**	50	50	(0)	74	47	**(-36)**	83	52	**(-37)**
K-NN	70	60	**(-14)**	70	73	(4)	82	73	**(-11)**	79	68	**(-14)**	75	61	**(-19)**	71	63	**(-11)**	82	70	**(-15)**	77	76	(-1)	77	63	**(-18)**	87	70	**(-20)**
KStar	76	62	**(-18)**	68	67	(-1)	71	56	**(-21)**	81	75	**(-7)**	71	62	**(-13)**	69	62	**(-10)**	72	66	**(-8)**	62	62	(0)	74	66	**(-11)**	83	76	**(-8)**
ADTree	78	72	**(-8)**	69	78	**(13)**	74	76	(3)	81	79	(-2)	76	71	**(-7)**	73	71	(-3)	85	82	(-4)	70	80	**(14)**	76	78	(3)	94	92	(-2)
J48	57	67	**(18)**	71	77	**(8)**	81	74	**(-9)**	76	77	(1)	78	68	**(-13)**	72	69	(-4)	90	80	**(-11)**	84	66	**(-21)**	78	78	(0)	93	91	(-2)
LMT	81	76	**(-6)**	76	80	(5)	78	77	(-1)	80	79	(-1)	71	76	**(7)**	72	71	(-1)	86	79	**(-8)**	83	79	(-5)	80	80	(0)	92	91	(-1)
Cart	74	59	**(-20)**	67	69	(3)	62	67	**(8)**	79	78	(-1)	67	62	**(-7)**	61	65	**(7)**	70	72	(3)	68	59	**(-13)**	63	66	(5)	79	81	(3)

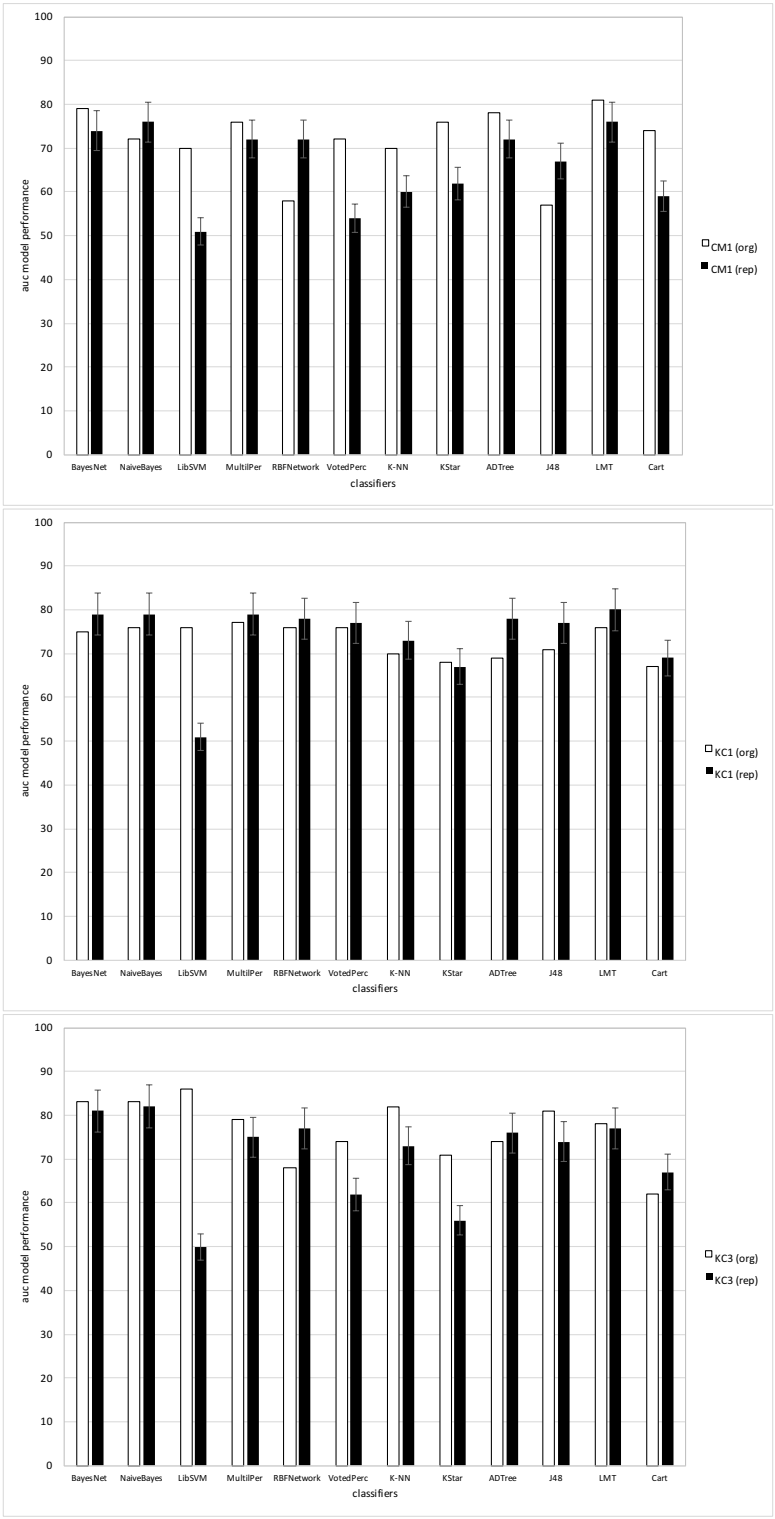
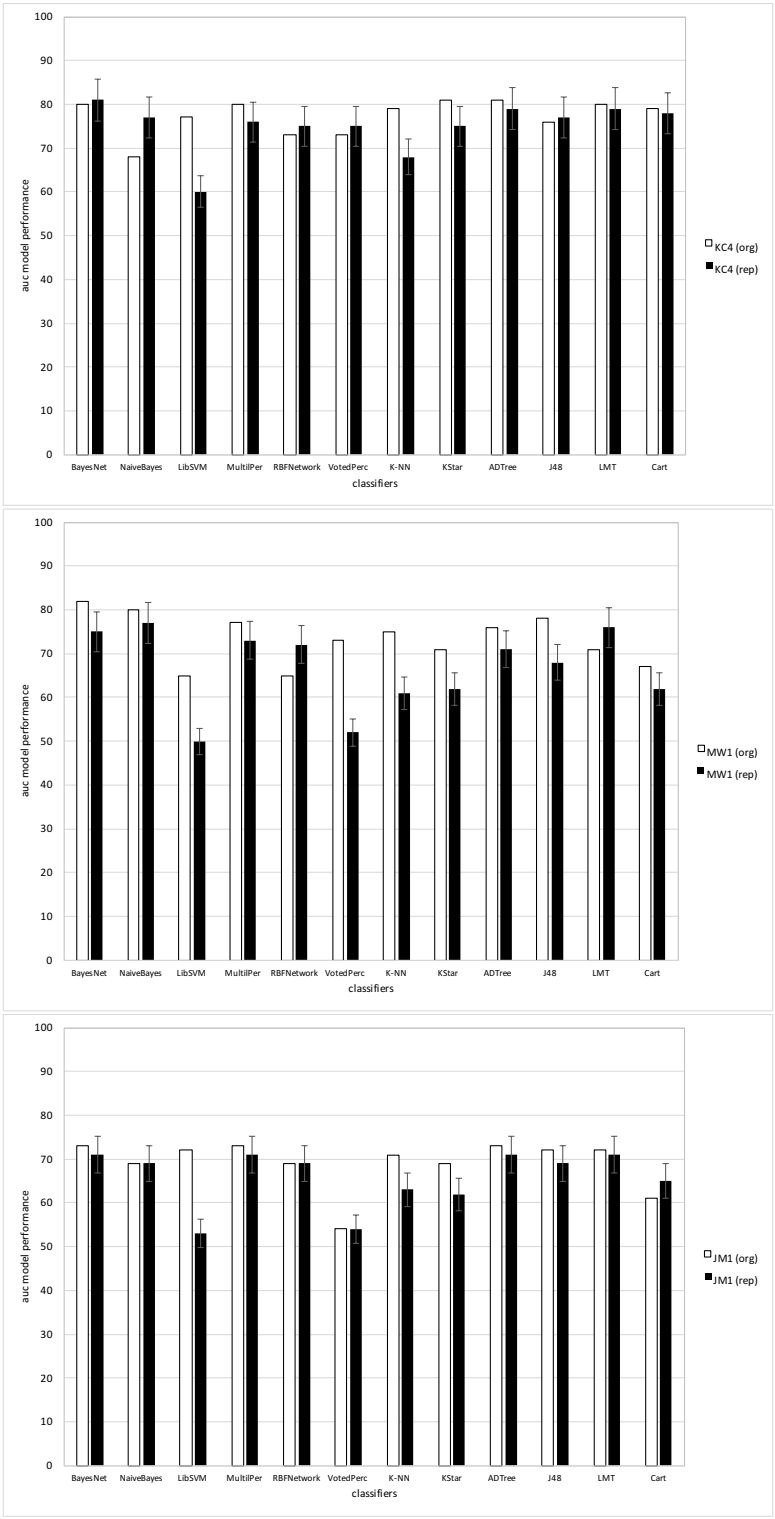
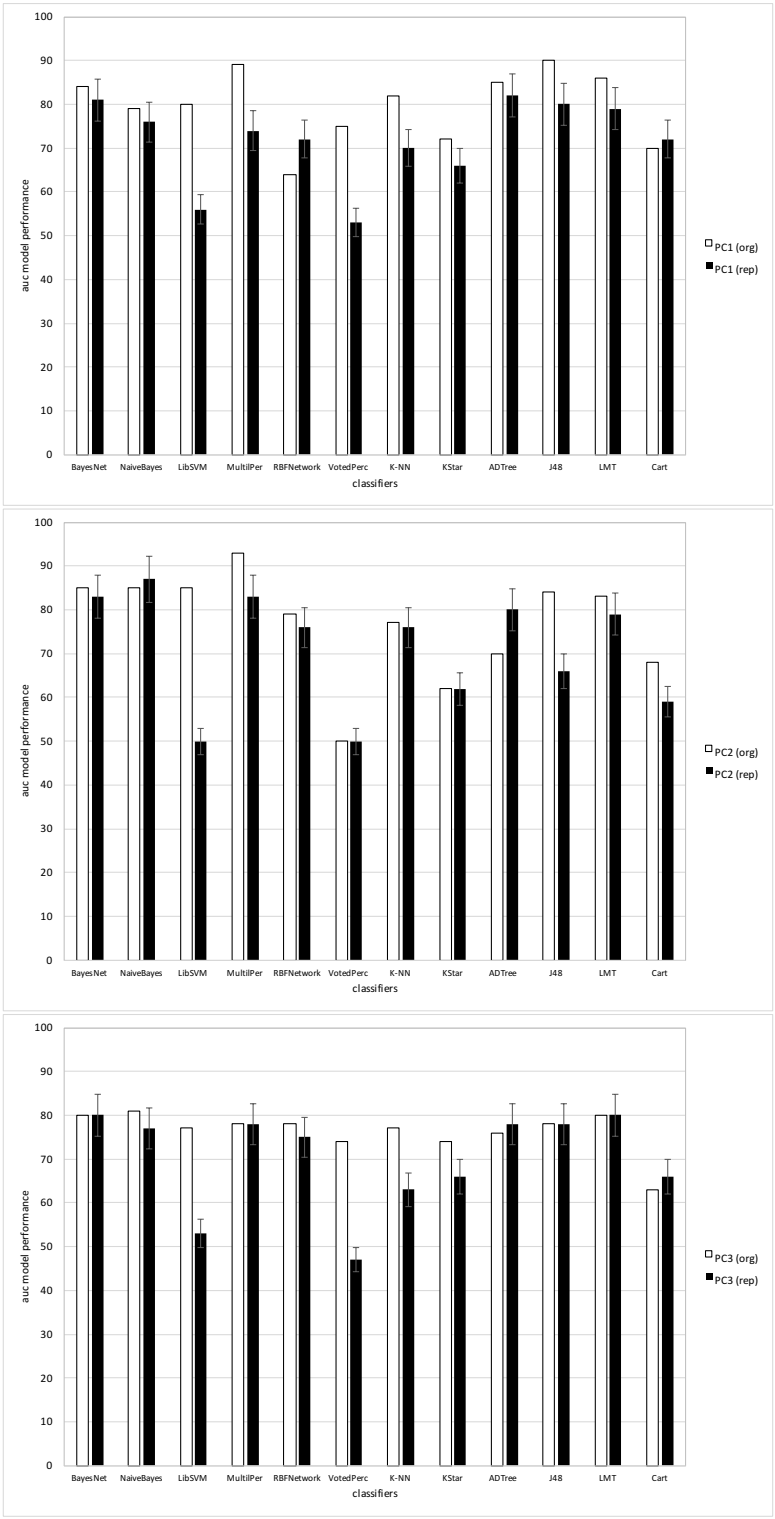


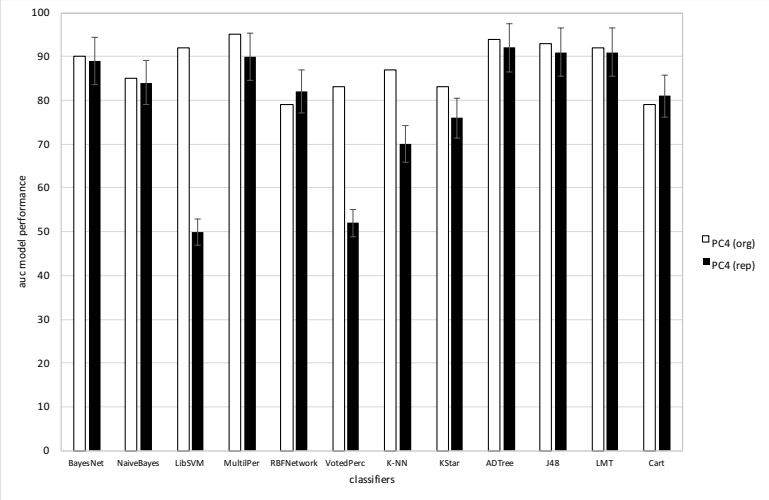
Figure 5.4: Bar plots showing the comparison of Lessmann et al. [2008] results for 12 classifiers on 10 datasets. The bars in white are the original results and in black are the reproduction results. In most cases the SVM and Voted Perceptron had the most differences over the datasets.



Bar plots showing the comparison of Lessmann et al. [2008] results for 12 classifiers on 10 datasets. The bars in white are the original results and in black are the reproduction results. In most cases the SVM and Voted Perceptron had the most differences over the datasets.



Bar plots showing the comparison of Lessmann et al. [2008] results for 12 classifiers on 10 datasets. The bars in white are the original results and in black are the reproduction results. In most cases the SVM and Voted Perceptron had the most differences over the datasets.



Bar plots showing the comparison of Lessmann et al. [2008] results for 12 classifiers on 10 datasets. The bars in white are the original results and in black are the reproduction results. In most cases the SVM and Voted Perceptron had the most differences over the datasets.

5.4 Reproducing the mining of Eclipse source code repository

The aim of Schröter et al. [2006] was to find out whether certain import relationships between imported classes or packages within a Java file can be used as input metrics to predict failures. That is to identify if a Java file is likely to fail when it imports a particular Java class or package. The study found out that certain classes or packages that appear in previous Java files that failed are useful for predicting failures. Some classes or packages are harder to work with. A Compiler package is harder to use than a GUI package which is easier and less likely to be prone to faults [Schröter et al. 2006].

Data: The study analysed Eclipse repository for release 2.0 and 2.1 and built prediction models. The study considered reported and fixed components that failed 6 months after their release to end-users.

Classifier: Four classifiers SVM, Ridge regression, Linear regression and Regression trees were used as both classifiers and rankers to carry out the prediction of post-release faults per Java file and package.

Tool: The experimental scripts were not provided. The authors reported that a syntactic tool to collect

the data from the repository was written but this tool was not available online.

Experimental Procedure of [Schröter et al. 2006]: Two repositories were used. A source code configuration management (CVS) that stores code changes, time, author, deletion, addition and modification. The second tool (BUGZILLA) is a bug database that stores reported bug summaries, status (i.e. assigned, fixed), bug ID, bug severity (trivial or major) etc. Fixed failures in BUGZILLA were mapped to the code that was changed to make the fix in CVS. The code changes occur in Java files of the Eclipse plugins analysed. For each Java file, its name, package name and the number of failures that occur post-release are recorded and provided as an XML document. Import statements were collected for each Java file and used as features to classify files as faulty or not and rank files as having the most faults.

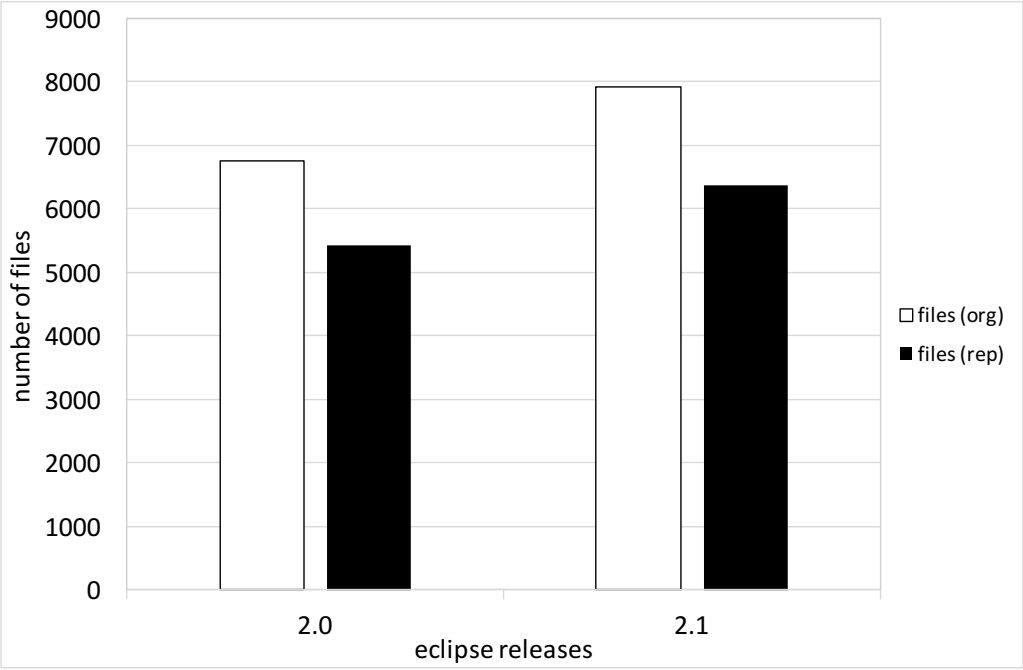
The training set in this case is 1/3 of Eclipse 2.0, an unusual training set size. In my previous experiments training sets were larger than testing sets for improved performance. Larger training sets are standard machine learning approaches [Hall et al. 2009]. 2/3 of Eclipse 2.0 and the whole of Eclipse 2.1 were used as the testing set. The experiment was repeated over 40 runs. Each run produced a prediction model trained on 1/3 train set. Trained models were tested separately on 2/3 test set of release 2.0 and the remaining 2.1. Averages over all runs and respective partitions were taken as final model performances. For classification a decision boundary was used. For example, files assigned probabilities above 0.5 are classified as faulty and below not faulty. For ranking the files are predicted based on the top 20%, 15%, 10%, 5% respectively. The performance of classifiers is based on precision (of all predictions made how many files were correctly classified) and recall (of all the actual failure prone files how many were correctly classified). Subsequently, for ranking (regression) Spearman rank correlation correlates the predicted number of faults in a Java file with the actual number of faults observed.

Reproducing [Schröter et al. 2006] and the changes made: The shared XML document by the original study contains the following information: linked bugs, their Java file and package names, number of

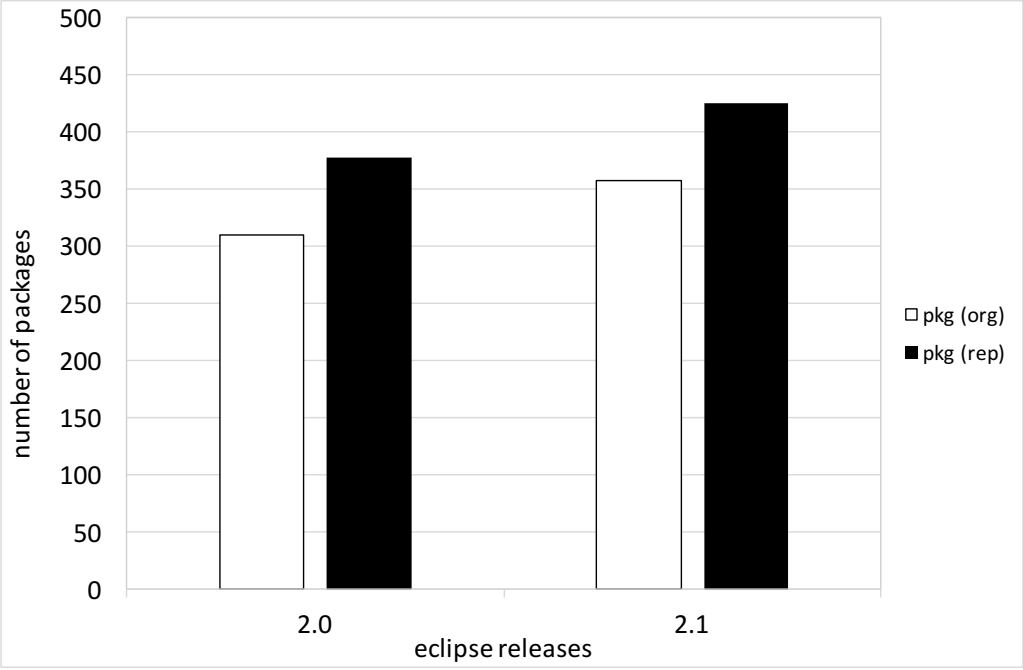
pre and post-release failures, and eclipse version. We (Mahmood, Bowes and Petric) wrote a tool, `EclipseCodeMiner`, in Java. It is used to convert the XML into Java objects which are output as comma separated values (CSV) with these headers (packagename, filename, pre and post release). The original study reported an analysis of 52 Eclipse plugins. For data consistency check, the tool extracted unique plugin names from the CSV file containing all plugins analysed by the original study. The extracted plugin names were used to copy only the exact plugin folders used by the original study. The original used the source code downloaded from the Eclipse repository. We also downloaded from the Eclipse repository. 42 plugins were found instead of the 52 plugins reported by the original; this is an indication that the dataset sizes will differ, thus affecting the prediction models. The tool was further used to collect all Java filenames from the 42 plugins and then reads all import statements in each of the Java files.

I then used R statistical package (version 3.4.0) to transpose the import statements used within each Java file. The transposed import statements become input metrics for each file. All import statements for all Java files are the feature-columns for the defect dataset created (i.e 3,421 features per file). If a file uses a set of import statements (represented as feature-columns in the defect dataset) a value of 1 is assigned for all members used by that Java file and 0 if not used. There is post release information, which is the number of failures a file has after being released. The post-release information is available in the XML output (now a CSV file). To create a complete defect dataset, I merged the post-release variable (or feature) with the transposed import statements by filename. The post-release variable serves as the dependent variable which the prediction models are able to predict.

Classification required the number of post-release failures to be a dichotomous dependent variable. Post-release values above 0 are flagged as 1 indicating the presence or absence of failures. Prediction by ranking uses the number of post-release failures as they are; this is because the model predicts the file with the highest number of failures.



(a) Java files mined from eclipse repository



(b) Java packages mined from eclipse repository

Figure 5.6: Schröter et al. [2006] results on mining a repository of 52 eclipse plugins. The difference in the number of Java files and packages of the plugins obtained by the original in white and reproduction study in black showing large differences in sizes of files and package for both studies. The original study reported mining of 52 plugins in the Eclipse repository, and reproducing the mining found 42 plugins. 10 plugins less than the original study, hence the discrepancy in number of files and packages.

Comparing the results: After creating the dataset, I did a data consistency check. A large discrepancy between plugins reported (52) in the original paper and those I downloaded (42) was observed. The result mean that thousands of Java files are missing. I provide the obtained results for this mining process in Figure 5.6. The Figure 5.6a shows the number of files reported in the original study in white and those our tool extracted in black. The left two bars show the number of Java files in thousands for release 2.0 of the eclipse system mined (the same goes for the bars on the right on release 2.1). The percentage difference in Table 5.5 shows the difference in files and packages mined from eclipse repository. The original study mined on 52 plugins has more files than my reproduction study mined on 42 plugins.

Table 5.5: The percentage difference between files and packages mined from eclipse repository for Schröter et al. [2006] and my reproduction study. Positive values in parentheses show that the my study has more files and negative ones show the number of packages are lower.

	files			packages		
versions	org	rep	diff	org	rep	diff
2.0	6,751	5,420	**(-20)**	309	377	**(22)**
2.1	7,909	6,369	**(-19)**	357	425	**(19)**

Table 5.5 and Figure 5.6b show number of packages in eclipse release 2.0 and 2.1 for the original and the reproduction study. My results are higher in both releases with 68 packages. It is strange that I obtained fewer files and higher number of packages than the original. It is possible that the 10 plugins missing in the same repository I mined affected the quantities. The 52 plugins used by the original contains fewer packages than those obtained in the 42 plugins I found in the eclipse repository. These results suggest that it may not matter which study obtained the larger number of plugins. Rather, it is which collection of plugins has the higher number of packages. The prediction models were not built for this study. Large changes in the dataset size affect the proportion of defective instances and model performance. My next reproduction study in section 5.6 shows this effect. To support my work I use 600 model results from [Shepperd et al. 2014] to assess the impact data imbalance has on model performance. The work is published in [Mahmood et al. 2015] and I give details of the results in chapter 6.

5.5 Reproducing a comparison of defect prediction approaches

The aim of [D'Ambros et al. 2010] is to produce a dataset that will be used for benchmarking prediction techniques. The focus was on the types of metrics used as features of defect datasets.

Data: There are five different types of metrics calculated from open source systems (Eclipse, Equinox, Mylyn, Pde, Lucene). The types of metrics include:

1. Change metrics calculated from CVS such as number of revisions made on a file and number of authors that worked on a particular change
2. Previous defects metrics calculated from number of past defect fixes
3. Source code metrics calculated as number of classes that reference a class
4. Entropy of changes metrics calculated from the distribution of one change in code that affects multiple files etc.
5. Churn of source code metrics calculated the number of code changes over a previous snap shot of the source code

Classifier: Linear regression was used to predict the number of post-release defects, a continuous dependent variable.

Tool: R statistical package was used.

Experimental procedure of [D'Ambros et al. 2010]: The defect dataset was first transformed (*pre-processed*) using Principal Component Analysis (PCA). Principal components that are independent and contribute at least 95% of variance within the dataset are selected as input to the prediction model; this prevents passing correlated metrics into the prediction model. 50 runs of randomised cross validation was then used to train the linear regression model on 90% training set and tested on 10% testing set. The final values are averages over the 50 runs. Adjusted R^2 is used to measure how well the regression

line fits the data with values from 0 to 1. The closer to 1 the more accurate the prediction model. Spearman rank correlation measures the similarity of the predicted ranking with the actual number of defects observed.

Reproducing [D'Ambros et al. 2010] and the changes made: The experiment required doing multiple runs for each type of metric. Performing data consistency checks revealed missing information which halted the reproduction study. The (CHGSET, change set size attributes) in (change-metrics.csv) was missing. Secondly, it was observed that all bugs are greater than the sum of the bug categories (non-trivial, major, critical and high priority) in the dataset. I contacted the original author, via email, to help clarify the data discrepancy but the author was not responsive for reasons unknown to me. Contacting authors has been investigated in the literature. It is a process of managing communication between researchers to achieve replications [Juristo et al. 2013]. Because D'Ambros et al. [2010]'s first reproduction and replication was found in chapter 4, we contacted [Mende 2010] and requested for any useful information. Mende [2010] sent the scripts used for his experiment.

Comparing the results: Mende [2010]'s script is reproducible, similar results between the original and the results from his script were observed. Table 5.6 shows Mende [2010]'s result and percentage difference values in parenthesis indicate the original study values are higher if positive or lower if negative. Despite the missing CHGSET metric in MOSER datasets, the results are very similar, indicating that that particular metric may not have a significant effect on the prediction model. Instead, few results still below 1% difference stand out for (Equinox, Pde, and Lucene systems). No explanation was found for these differences, however, they are below 1%. It is still important to do data consistency checks. The missing feature discovered did not have an effect on model performance. The CHGSET metric may be highlighted as not very useful for prediction.

Table 5.6: Mende script results compared in % difference showing D'ambros experiments are reproducible

	Adjusted R2 - Explanative power					Spearman's - Predictive power				
	eclipse	mylyn	equinox	pde	lucene	eclipse	mylyn	equinox	pde	lucene
MOSER	0.460 (0.006)	0.210 (0.004)	0.612 (0.016)	0.517 (0.000)	0.602 (0.032)	0.419 (0.096)	0.328 (0.044)	0.503 (-0.031)	0.246 (0.081)	0.226 (-0.012)
NFIXONLY	0.144 (0.001)	0.043 (-0.000)	0.425 (0.004)	0.139 (0.001)	0.400 (0.002)	0.330 (0.042)	0.145 (-0.003)	0.458 (0.029)	0.138 (0.025)	0.234 (-0.050)
NR	0.379 (-0.001)	0.125 (-0.003)	0.524 (0.004)	0.365 (-0.000)	0.489 (0.002)	0.404 (0.040)	0.096 (-0.003)	0.581 (0.033)	0.265 (0.020)	0.257 (-0.039)
NFIX_NR	0.383 (-0.000)	0.126 (-0.003)	0.525 (0.004)	0.365 (-0.000)	0.488 (0.029)	0.404 (0.023)	0.110 (0.019)	0.582 (0.015)	0.266 (0.011)	0.254 (-0.023)
BF	0.456 (-0.031)	0.129 (-0.032)	0.476 (-0.027)	0.533 (-0.006)	0.559 (-0.000)	0.447 (0.037)	0.117 (-0.042)	0.538 (0.046)	0.282 (0.003)	0.318 (-0.059)
BUGCAT	0.489 (0.034)	0.160 (0.029)	0.517 (0.048)	0.536 (-0.003)	0.559 (0.000)	0.428 (-0.006)	0.240 (0.109)	0.514 (0.001)	0.275 (-0.009)	0.318 (-0.035)
CK	0.385 (-0.034)	0.115 (-0.080)	0.564 **(-0.109)**	0.061 **(-0.573)**	0.378 (-0.001)	0.401 (0.011)	0.276 (-0.023)	0.535 (0.082)	0.297 (0.013)	0.181 (-0.033)
OO	0.410 (0.028)	0.169 (0.054)	0.622 (0.065)	0.606 **(0.548)**	0.236 **(-0.132)**	0.416 (0.039)	0.320 (0.094)	0.551 (0.067)	0.293 (0.037)	0.152 (-0.064)
CK_OO	0.426 (0.020)	0.195 (0.025)	0.679 (0.060)	0.622 (0.004)	0.391 **(0.182)**	0.432 (0.037)	0.306 (0.009)	0.488 (-0.002)	0.310 (0.047)	0.171 (-0.043)
LOC	0.353 (0.005)	0.040 (0.001)	0.415 (0.007)	0.042 (0.002)	0.083 (0.006)	0.406 (0.026)	0.238 (0.016)	0.504 (0.029)	0.271 (0.021)	0.129 (-0.043)
HCM	0.364 (-0.002)	0.025 (0.001)	0.497 (0.002)	0.132 (0.002)	0.308 (0.000)	0.462 (0.046)	-0.004 (-0.003)	0.571 (0.045)	0.261 (0.017)	0.266 (-0.042)
WHCM	0.376 (0.003)	0.038 (-0.000)	0.345 (0.005)	0.168 (0.003)	0.492 (0.002)	0.447 (0.046)	0.056 (-0.020)	0.563 (0.030)	0.289 (0.016)	0.260 (-0.028)
EDHCM	0.209 (-0.000)	0.026 (0.000)	0.352 (0.007)	0.256 (0.003)	0.222 (0.002)	0.378 (0.007)	0.060 (-0.010)	0.537 (0.042)	0.266 (0.008)	0.264 (-0.042)
LDHCM	0.161 (0.000)	0.011 (-0.000)	0.468 (0.005)	0.270 (0.003)	0.218 (0.002)	0.383 (0.006)	0.054 (-0.010)	0.559 (-0.022)	0.290 (0.010)	0.264 (-0.011)
LGDHCM	0.054 (-0.000)	0.001 (0.001)	0.511 (0.003)	0.212 (0.003)	0.146 (0.005)	0.395 (0.031)	0.032 (0.002)	0.578 (0.016)	0.276 (0.013)	0.282 (-0.048)
CHU	0.452 (0.007)	0.170 (0.001)	0.657 (0.012)	0.616 (-0.012)	0.481 (0.025)	0.392 (0.021)	0.201 (-0.025)	0.425 (-0.085)	0.248 (-0.003)	0.192 **(-0.100)**
WCHU	0.516 (0.004)	0.191 (-0.000)	0.656 (0.011)	0.601 (-0.007)	0.510 (0.032)	0.434 (0.015)	0.260 (-0.019)	0.518 (-0.042)	0.280 (0.002)	0.220 (-0.065)
LDCHU	0.562 (0.005)	0.214 (0.000)	0.598 (0.017)	0.615 (-0.001)	0.515 (0.057)	0.412 (0.017)	0.227 (-0.048)	0.515 (-0.048)	0.321 (0.014)	0.257 (-0.036)
EDCHU	0.515 (0.006)	0.227 (0.000)	0.540 (0.015)	0.599 (0.001)	0.527 (0.060)	0.379 (0.017)	0.238 (-0.021)	0.475 (0.011)	0.308 (0.014)	0.249 (-0.031)
LGDCHU	0.479 (0.006)	0.099 (0.004)	0.653 (0.011)	0.500 (0.014)	0.512 (0.019)	0.401 (-0.041)	0.186 (-0.002)	0.511 (-0.055)	0.273 (0.084)	0.243 (-0.047)
HH	0.486 (0.002)	0.198 (-0.001)	0.675 (0.008)	0.519 (0.005)	0.482 (0.049)	0.433 (0.028)	0.267 (-0.010)	0.544 (0.060)	0.266 (-0.000)	0.188 **(-0.130)**
HWH	0.480 (0.007)	0.147 (0.001)	0.635 (0.014)	0.630 (-0.011)	0.506 (0.022)	0.408 (-0.017)	0.169 (-0.043)	0.451 (-0.029)	0.265 (-0.001)	0.219 (-0.044)
LDHH	0.535 (0.004)	0.208 (-0.001)	0.608 (0.012)	0.534 (0.012)	0.433 (0.090)	0.411 (0.003)	0.246 (-0.026)	0.561 (0.031)	0.307 (0.011)	0.212 (-0.121)
EDHH	0.489 (0.004)	0.225 (-0.001)	0.486 (0.017)	0.527 (0.012)	0.442 (0.083)	0.380 (0.014)	0.254 (-0.019)	0.522 (-0.064)	0.315 (0.011)	0.211 (-0.126)
LGDHH	0.481 (0.002)	0.132 (0.002)	0.669 (0.009)	0.466 (0.019)	0.452 (0.033)	0.424 (0.003)	0.203 (0.018)	0.549 (0.057)	0.263 (0.027)	0.231 (-0.116)
BF_CK_OO	0.496 (0.004)	0.212 (-0.001)	0.713 (0.006)	0.637 (-0.012)	0.588 (0.002)	0.450 (0.011)	0.292 (0.015)	0.525 (-0.022)	0.306 (0.024)	0.298 (-0.064)
BF_WCHU	0.540 (0.004)	0.193 (0.000)	0.656 (0.011)	0.618 (-0.009)	0.596 (0.002)	0.441 (-0.007)	0.243 (-0.022)	0.510 (-0.023)	0.275 (-0.007)	0.274 (-0.036)
BF_LDHH	0.565 (0.004)	0.216 (-0.001)	0.626 (0.011)	0.603 (0.002)	0.595 (0.003)	0.422 (0.000)	0.219 (-0.002)	0.550 (0.017)	0.281 (-0.024)	0.276 (-0.076)
BF_CK_OO_WCHU	0.496 (-0.063)	0.212 (-0.038)	0.713 (-0.021)	0.637 (-0.024)	0.588 (-0.022)	0.450 (0.025)	0.292 (-0.014)	0.525 (0.001)	0.306 (-0.004)	0.298 (-0.000)
BF_CK_OO_LDHH	0.591 (0.004)	0.259 (-0.003)	0.739 (0.009)	0.671 (-0.009)	0.621 (0.003)	0.432 (-0.008)	0.280 (-0.011)	0.455 **(-0.116)**	0.321 (0.009)	0.275 **(-0.102)**
BF_CK_OO_WCHU_LDHH	0.591 (-0.029)	0.259 (-0.018)	0.739 (-0.015)	0.671 (-0.020)	0.621 (-0.029)	0.432 (0.024)	0.280 (-0.046)	0.455 **(-0.137)**	0.321 (0.032)	0.275 (-0.066)

5.6 Reproducing defect prediction approach on Eclipse

The aim of Zimmermann et al. [2007]'s study is to map defects reported in Eclipse bug database to the source code in order to find the most defect prone components of a software.

Data: The same Eclipse system (releases 2.0 and 2.1) used by Schröter et al. [2006] was used with an additional (release 3.0). In this extended defect dataset, complexity metrics were calculated using Eclipse and used to find out if more complex code has more bugs. More metrics were calculated based on the abstract syntax tree (AST). Each Java file or package had the AST nodes (e.g. *ForStatement, IfStatement*) as their columns. Frequency of the nodes become the values in the dataset. Files with 1 or more failures post release are flagged with 1 for defective or 0 for non defective.

Classifier: Using logistic regression the classification model predicts for each file a likelihood value between (0 and 1). Predicted likelihood values of files above 0.5 were classified as faulty, anything less is not faulty. Precision, recall and accuracy[3] measures were used to evaluate the classification model. Predicting defects by ranking was done with linear regression to predict the number of post release defects per file or package. Ranked predictions were compared with the actual ranks using Spearman correlation.

Tool: R statistical package was also used here.

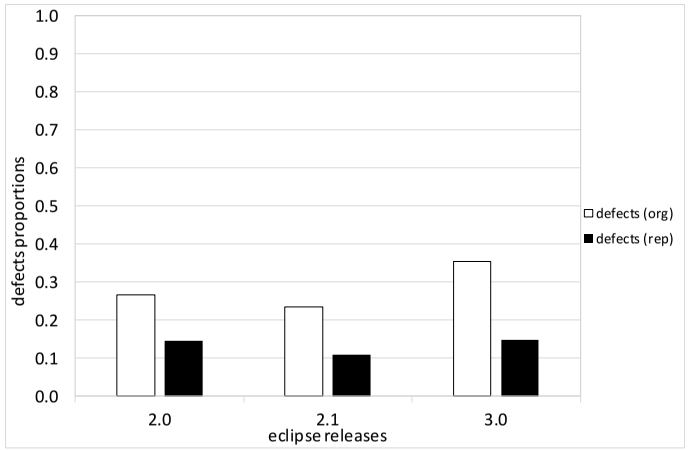
Reproducing [Zimmermann et al. 2007] and the changes made: The authors provided all scripts and datasets for the study. The script is runnable but the results are not reproducible. Performing the data consistency check showed large variations in dataset sizes. The sizes vary with what was reported in the paper against what I downloaded from their research group data repository. On the website the authors reported "*Regrettably, due to a clerical error, we have temporarily withdrawn all versions of*

[3]note that these measures are not comprehensive like Mathews correlation coefficient which assesses the model based on correctly/incorrectly classified majority and minority classes

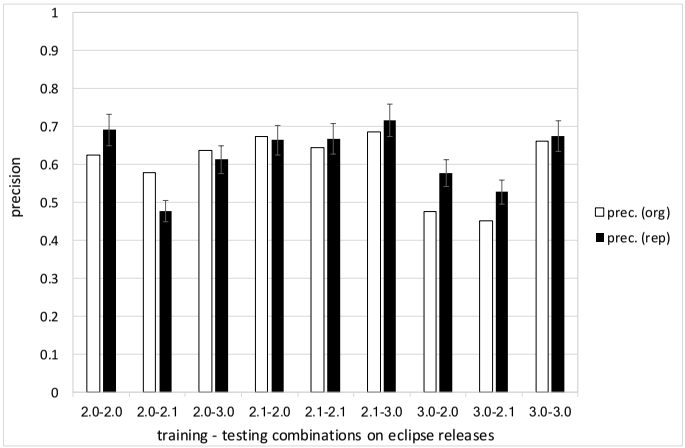
the Eclipse bug data..We removed duplicates entries from the dataset that were inserted by a bug in the Perl API. Here's the new data (labelled as Release 2.0a)"[4]. A new version of the data is now available, however, would this lead to a rework or withdrawal of the initial paper? This question is interesting but not answerable within the scope of my work.

Comparing the results: Figure 5.7 shows large variations in the results after running the classification experiment compared to the original study. Figure 5.8a shows significant reductions between the first (in white) and second (in black) versions of the defect datasets. The number of defects per release has drastically reduced and is likely to affect the models. The reduction of defects affects the balance between defective and non defective files leading to the imbalanced data problem; this problem affects prediction models (e.g. [Van Hulse et al. 2007, Lane et al. 2012]). More about the data imbalance problem is discussed in chapter 6. The effect of this change is noticeable in the rest of the figures. Figure 5.8b shows my results are similar in 5 instances and not in 4 instances. The $x - axis$ represents training and testing combinations for the model. The first data point is the model trained on release 2.0 and tested on release 2.0. Subsequent train-test combinations are: 2.0-2.1, 2.0-2.1, 2.0-3.0, 2.1-2.0, 2.1-2.1, 2.1-3.0, 3.0-2.0, 3.0-2.1, 3.0-3.0 respectively. The number of defects in my data is fewer than that of the original. My model correctly classified the defective instances it predicted (higher precision than original). It does not necessarily mean that the model correctly predicted all defective instances. In fact in Figure 5.8c the model missed out many defective instances (lower recall than the original). We are more interested in finding defects, Figure 5.8d shows higher accuracy than the original study. The accuracy is higher because of the fewer defective instances in the version of the dataset I used. Accuracy performance goes high if the majority class is dominant in the data. The accuracy measure is not suitable to measure model performance on imbalanced data. Since the model missed most of the defective instances and got the non defective ones it has low recall and high accuracy.

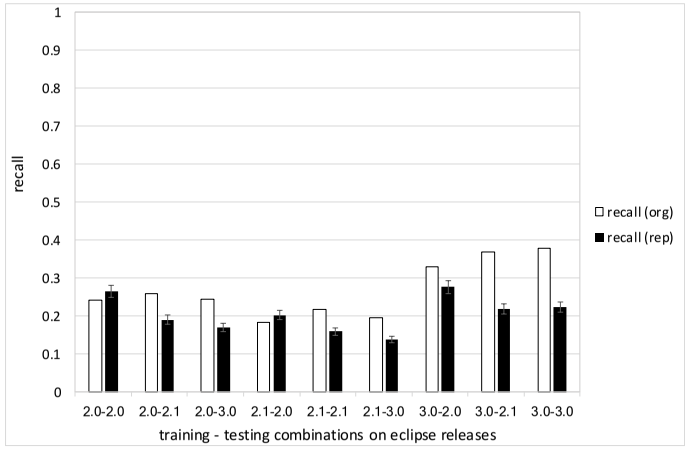
[4]https://www.st.cs.uni-saarland.de/softevo/bug-data/eclipse/: (Accessed on 18th October 2017)



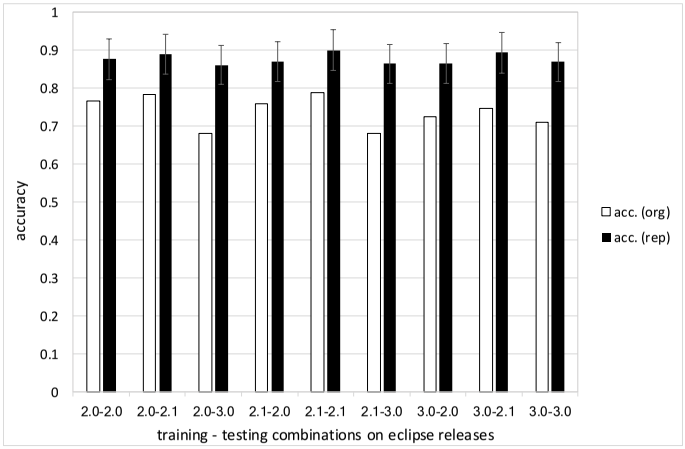
(a) data variation



(b) precision



(c) recall



(d) accuracy

Figure 5.7: Zimmermann et al. [2007] script results compared in % difference showing most experiments are not reproducible due to large variation in the datasets. Train-test combinations show the model is trained on release 2.0 and tested on release 2.0, subsequently on (2.0-2.1, 2.0-2.1,2.0-3.0, 2.1-2.0, 2.1-2.1, 2.1-3.0, 3.0-2.0, 3.0-2.1, 3.0-3.0) respectively.

Table 5.7: The percentage difference of Zimmermann et al. [2007] results and my reproduction study. The positive (diff) values indicate my reproduction has higher values than the original. Majority of the differences in the datasets combinations (left) and model performance measures are greater than 5%. This means the study is not reproducible due to the change in defect proportions.

	defects				precision			recall			accuracy		
versions	org	rep	diff	train-test	org	rep	diff	org	rep	diff	org	rep	diff
2.0	0.27	0.14	**(-45)**	2.0-2.0	0.625	0.692	**(11)**	0.242	0.265	**(9)**	0.766	0.876	**(14)**
2.1	0.23	0.11	**(-54)**	2.0-2.1	0.578	0.478	**(-17)**	0.259	0.191	**(-26)**	0.783	0.890	**(14)**
3.0	0.36	0.15	**(-58)**	2.0-3.0	0.638	0.613	(-4)	0.244	0.171	**(-30)**	0.682	0.861	**(26)**
				2.1-2.0	0.673	0.664	(-1)	0.185	0.203	**(10)**	0.760	0.870	**(14)**
				2.1-2.1	0.645	0.668	(4)	0.219	0.160	**(-27)**	0.789	0.900	**(14)**
				2.1-3.0	0.687	0.717	(4)	0.195	0.139	**(-29)**	0.682	0.864	**(27)**
				3.0-2.0	0.476	0.578	**(21)**	0.330	0.277	**(-16)**	0.726	0.866	**(19)**
				3.0-2.1	0.453	0.528	**(17)**	0.369	0.220	**(-40)**	0.748	0.894	**(20)**
				3.0-3.0	0.663	0.675	(2)	0.379	0.224	**(-41)**	0.711	0.869	**(22)**

5.7 Conclusion

In this chapter I have reproduced five studies (four failed) and identified potential factors that may affect model performance; this partially answers my (RQ5).

Table 5.8: The five studies I reproduced compared based on their experimental procedure. Important methodological aspects of defect prediction studies are also considered. A reproducibility assessment outcome indicator showing whether the results are reproducible or not. All studies have cross validation in common and all studies have data problems (imbalance, cleaning, consistency). Only one study is fully reproducible.

	Experimental procedure components & outcomes									
Original studies	Normalisation	Cross Validation	Imbalance	Data Cleaning	Data Consistency	Tuning	Feature Selection	Experimental Script	Reproducible	Replicable
Menzies et al. [2007]	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗
Schröter et al. [2006]	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓
D'Ambros et al. [2010]	✗	✓	✗	✗	✗	✗	✗	✓	✓	✓
Zimmermann et al. [2007]	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗

Overall this chapter provides empirical evidence that attempting to use the same data, tools, and method, does not guarantee the reproducibility of a study (Table 5.8). That is, a study may still not be verifiable through this approach. Even though the study has been replicated and confirmed with different permutations of data, tools and method. This 'discrepancy' is based on the differences in measuring reproducibility and replicability. Otherwise the results may seem to contradict each other where a study is replicable but not reproducible. I note recent concerns about non usage of robust statistical methods to confirm findings [Madeyski and Kitchenham 2017]. Perhaps this may have affected statistically confirmed claims in the original and its replication study (that confirmed the original's finding). Further works on how to measure replications are needed.

Looking at Table 5.8, all studies I reproduced are affected by dataset problems. Only one study, D'Ambros et al. [2010] , is fully reproducible and replicable. Interestingly, the data used by D'Ambros et al. [2010] has missing metrics which affected my results. The missing metrics was confirmed by Mende [2010] who also reproduced the original. However, when I ran the script by Mende [2010] the results was close to the original. I could not explain this anomaly, since there is missing data and results are the same as the original. Could there be a possibility that Mende [2010] performed additional unreported tasks to achieve the same result as the original, or the results are random. Perhaps this is the major threat identified in this chapter. When problems in original studies are discovered, these studies are meant to be corrected. When corrected, should the original studies remain in the search space? Or should they be flagged at the point of access (in the online library) as having been challenged, possibly flawed? This is so that researchers learn from the errors of such studies, treat them with caution, and incorporate the corrections. Otherwise the errors in those studies may continue to propagate across the field, potentially, compromising the validity of findings. Furthermore, in Table 5.8, dealing with data imbalance, data cleaning, and parameter tuning are not dealt with. I believe this is very serious, suboptimal models may have been designed over the years due to poor methodology. There is therefore, a lot of factors that need to be addressed in defect prediction. There is opportunity for improving these studies by ensuring all these aspects of the procedure where needed are adhered to.

In addition, the lack of a single version of data may be a significant factor that affects reproducibility but by how much is still unknown. To aid reproducibility a single version of data between an original and the repository should be maintained. Tools with their versions need to be reported. Parameters of algorithms both input and those outputted need to be reported. My reproduction results are mixed. There could be many unknown factors that could have caused these differences in the model performances. Not all factors can be investigated in this work. In chapter 6 I investigate what impact variations in data, tool versions, and unreported preprocessing steps have on model performance.

6. Investigations of Factors Affecting Reproducibility

6.1 Introduction

The results of chapter 5 have shown the likely factors affecting reproducibility of a study. The factors are data inconsistency, missing tool versions and data preprocessing steps during prediction modelling. Dataset information in original papers has not been consistent with the information in the online repository. When there is a large difference between the sizes of different versions of the same datasets (on paper and in repository), the number of defective and non defective modules are also different. Prediction model performance on these datasets tend to be affected greatly on both datasets.

In one of my reproduction experiments, all the modelling information was provided, yet the reproduction failed the most on two classifiers. I use WEKA for all the experiments I performed. Instances where I use the same tool as the original still showed different results. Therefore, I consider the possibility that versions of tools may give different results.

The aim of this chapter is to complete the results necessary to answer (RQ5). That is whether replicated studies can be reproduced and to identify factors that aid or hinder reproduction. I do more experiments to validate effects of dataset inconsistency, data preprocessing, and consistency of WEKA on model performance. These factors need to be validated to guide future researchers on the impact they have on model performance and how they hinder reproducibility – a means of checking research validity.

I start with the data inconsistency problem. I statistically analyse two factors to determine the impact

'change in defect proportions' has on 'model performance'. A sufficient sample size to do a reliable statistical test is needed. For that, I use data synthesised by a previous meta-analysis of 600 defect prediction models [Shepperd et al. 2014]. I then use the computed Mathews Correlation Coefficient (*MCC*) model evaluation measure and compare it to the corresponding change in defect proportions (also known as data imbalance ratio in the literature). These two variables also show how defect prediction models perform and how data imbalance affects the prediction performance.

Secondly, I investigate the overwhelming non-reproducibility of the two classifiers in chapter 5, Support Vector Machine (SVM) and Voted Perceptron (VP) in my reproduction study of Lessmann et al. [2008]. I then rescale the dataset by taking the natural log of its values. The scaling is done before building the prediction models (a preprocessing step); this step has been used originally by [Menzies et al. 2007] to improve model performance. Lessmann et al. [2008] replicated that original study but did not report whether they used the log transform or not.

Most of the originals did not report versions of the tools they use. Finally, I analyse different versions of WEKA. I present my results in the following format: the aim, data, tools, experimental procedure, and results.

6.2 Effect of Imbalanced Data on Model Performance and Reproducibility

The aim of this study is to test whether the change in the balance of a dataset affects predictive performance significantly. Secondly, to check whether moving to a more balanced dataset increases that performance. The outcome would confirm that dataset inconsistency is a threat to reproducibility. It will also confirm that model performance may increase significantly. The work in this section has been published [Mahmood et al. 2015]. I was the lead-author of this conference paper, along with 3 co-authors.

My role was to conduct the initial analysis on the Shepperd et al. [2014] dataset and write the paper. Subsequently, the study was validated by each co-author for each of the 3 research aspects. The aspects include statistical analysis, dataset imbalance problem, and a review and validation of the whole study.

Data: The datasets from Shepperd et al. [2014] are based on defect prediction studies which have predicted a module of code as being either defective or not defective. When a model is trained to predict an instance as being defective or not defective the results can be summarised in a confusion matrix (see Table 2.4 in chapter 2). Compound performance measures can be computed from the confusion matrix to reveal different properties of the prediction models (see Table 2.5 in chapter 2). The following are standard classification measures, MCC, $F-Measure$, $Precision$ and $Recall$, extracted from confusion matrices. The MCC measure was chosen because it combines all four quadrants of the confusion matrix and does not ignore the many true negatives; $Precision$ because high values indicate that few predictions are wrong and a developer would not be wasting their time investigating these predictions; $Recall$ shows the proportion of defects actually predicted, which is important in safety critical systems; and $F-measure$ combines both $Precision$ and $Recall$.

Tool: R statistical software (3.1.3) is used.

Experimental procedure: The percentage of the defective class (*balance%*) of the datasets used for the 600 models in Shepperd et al. [2014] is extracted. A significant impact on predictive performance due to a change in *balance%* should confirm my hypothesis. That, data inconsistencies (discovered in chapter 5 between original paper and repository) may have affected my reproduction experiments. However, data inconsistency may not be the only factor. An analysis of variance of the Shepperd et al. [2014] data is done. The numerical *balance%* is converted into 20 factor levels ($balance_5$) e.g. factor 1 is any imbalance in the range 0.0% to 4.9%.

Results: Table 6.1 shows that $balance_5$ does contribute significantly to the variance in the results, how-

ever, dataset family (from which imbalance is derived) contributes more. In conclusion, imbalance of the dataset is important, but dataset family has other features (beyond the scope of this study) which are causing more variations in the performance of defect prediction studies.

Table 6.1: Table showing how the variance result from $balance_5$ compares with other possible factors.

	Partial η^2	Pr(>F)
Researcher Group	31.01%	< 0.0001
Dataset Family	31.00%	< 0.0001
$balance_5$	**18.76%**	< 0.0001
Input Metrics	12.44%	< 0.0001
Classifier Family	8.23%	< 0.0001

Having established that data imbalance can be a significant factor in defect prediction, the nature of the relationship between Predictive performance and imbalance is now investigated. Scatterplots of $F-measure$, $Precision$ and $Recall$ against $balance\%$ are created.

Figure 6.1 shows that, as $balance\%$ initially increases, MCC also increases. This means that datasets with very few defects tend to result in prediction models which are poor at predicting defects. There is a change in MCC from 0.15 to 0.35 when $balance\%$ changes from 0.00 to 0.20. Figure 6.1 has few data points where $0.21 < balance\% < 0.30$ and, although average predictive performance appears to increase, we can not be confident in saying that performance increases up to $balance\%$=0.3. The general increase is also observed for $F-Measure$ and $Precision$. It is interesting to note that $Recall$ initially decreases as $balance\%$ increases. As $balance\%$ increases from 0.3 to 0.5 (maximum level of balance), the predictive performance does not tend to increase for MCC. Predictive performance as measured by $Precision$, $Recall$ and $F-Measure$ do increase.

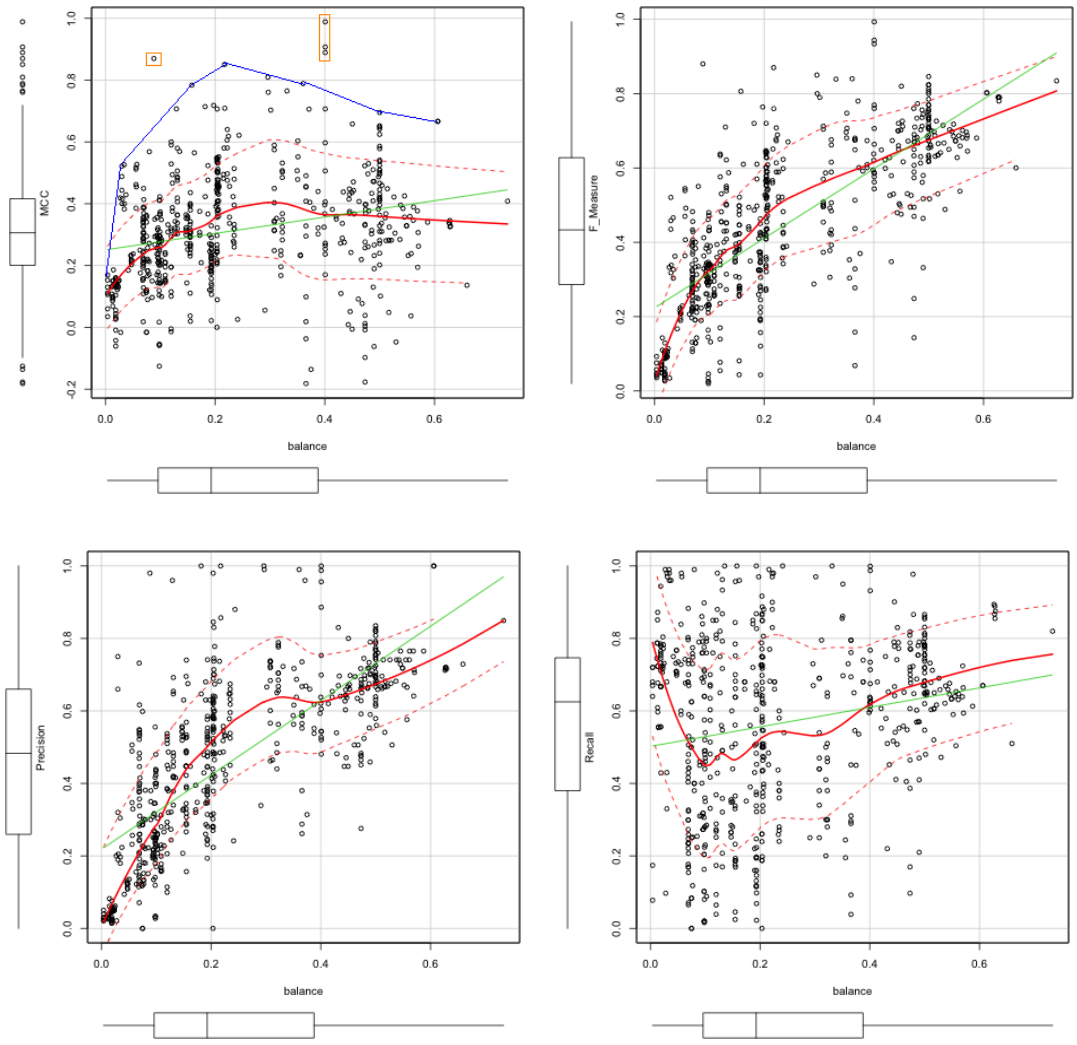


Figure 6.1: Scatterplot of predictive performance against dataset *balance%*. Showing that for *MCC*, as the *balance%* increase from 0.00 to 0.20, the *MCC* increases. An increase of *balance%* from 0.20 to 0.5 does not seem to change *MCC*. For the first scatterplot, we identify a small number of possible outliers (orange boxes). We have also plotted the convex-hull of 'best' performance (blue line) as *balance%* increases. The (red lines) indicate the overall moving average trend of the performance against the balance of the data. The (dotted red lines) indicate the lowest and highest average of the trend. The (green lines) are linear trend lines, all showing an increase in performance as the balance increases.

In light of reproducibility, this novel finding about how imbalanced data impacts predictive performance may explain some of the reproducibility failures I encountered in chapter 5. Major rewrites of datasets affects the balance of the data, and affected balance implies a significant change in the predictive performance. Therefore, part of reproducibility success depends upon a single version of dataset consistent in the original paper and available online. I now investigate additional factors affecting reproducibility even after matching all parameters, metrics and most of the datasets in [Lessmann et al. 2008].

6.3 Effect of Data Preprocessing

As reported earlier in chapter 5 my reproduction experiment of [Lessmann et al. 2008] of 12 classifiers was not successful especially for two classifiers. The model performance averages for both original and my reproduction study are: SVM (79 and 53) and for VP (72 and 56) respectively. These differences could not be explained. The aim is to test whether there are missing preprocessing steps not reported by Lessmann et al. [2008].

Data: 10 NASA datasets were used as mentioned in chapter 5 Table 5.4. There are few data inconsistencies between the dataset information in original paper and repository.

Tools: WEKA 3.9.1 developer was used.

Experimental procedure: The natural log was taken for each value of each feature of the 10 datasets for scaling and improving model performance. For example a feature such as lines of code for a small and large Java file (module) can be (10 and 720, taking the $ln(10)$ $ln(720)$ –> $2.3, 6.57$). The log filter I used is the same filter used by [Menzies et al. 2007]. Lessmann et al. [2008] replicated the same study before producing their novel framework which I reproduced. I assume that the preprocessing step might have been used but not reported. The rest of the experiment is the same as the original, but I ran it 1,000 by 10 fold cross validation for increased reliability of model performance as mentioned in chapter 5.

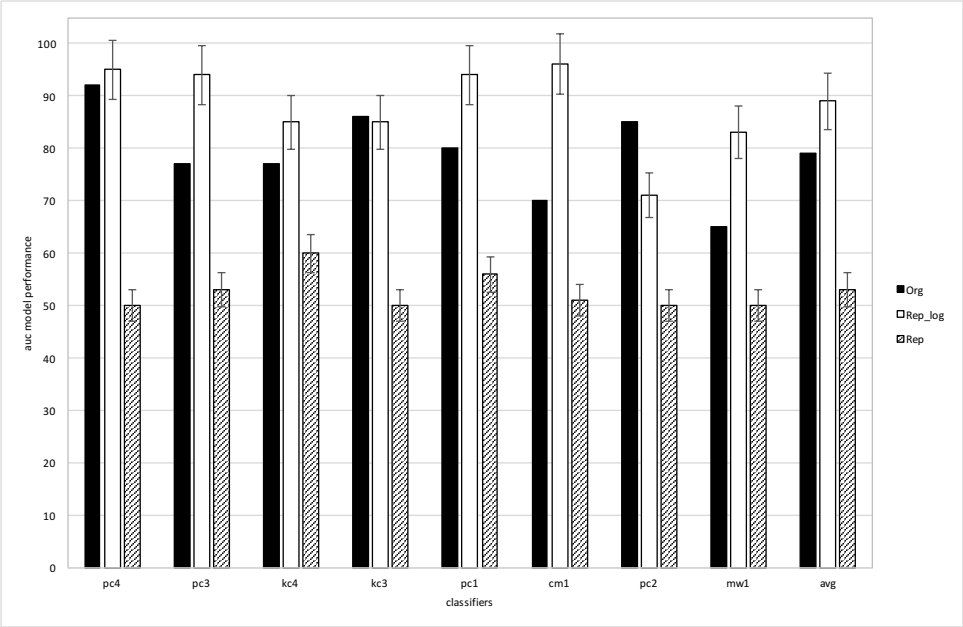
Table 6.2: A comparison of Lessmann et al. [2008] results and my reproduction experiments run twice, with and without log transforming the dataset values. SVM and VP classifiers show poor performance compared to the original and increase significantly after applying the log transform. Positive percentage difference in parenthesis indicates that the my results are higher than the original and vice versa.

	SVM					VP				
Datasets	Org	Rep(*log*)	diff(*log*)	Rep	diff	Org	Rep(*log*)	diff(*log*)	Rep	diff
pc4	92	95	(3)	50	**(-46)**	83	87	(5)	52	**(-37)**
pc3	77	94	**(22)**	53	**(-31)**	74	74	(0)	47	**(-36)**
kc4	77	85	**(10)**	60	**(-22)**	73	79	**(8)**	75	(3)
kc3	86	85	(-1)	50	**(-42)**	74	83	**(12)**	62	**(-16)**
pc1	80	94	**(18)**	56	**(-30)**	75	79	(5)	53	**(-29)**
cm1	70	96	**(37)**	51	**(-27)**	72	79	**(10)**	54	**(-25)**
pc2	85	71	**(-16)**	50	**(-41)**	50	50	(0)	50	(0)
mw1	65	83	**(28)**	50	**(-23)**	73	77	(5)	52	**(-29)**
avg	79	89	**(13)**	53	**(-33)**	72	76	**(6)**	56	**(-22)**

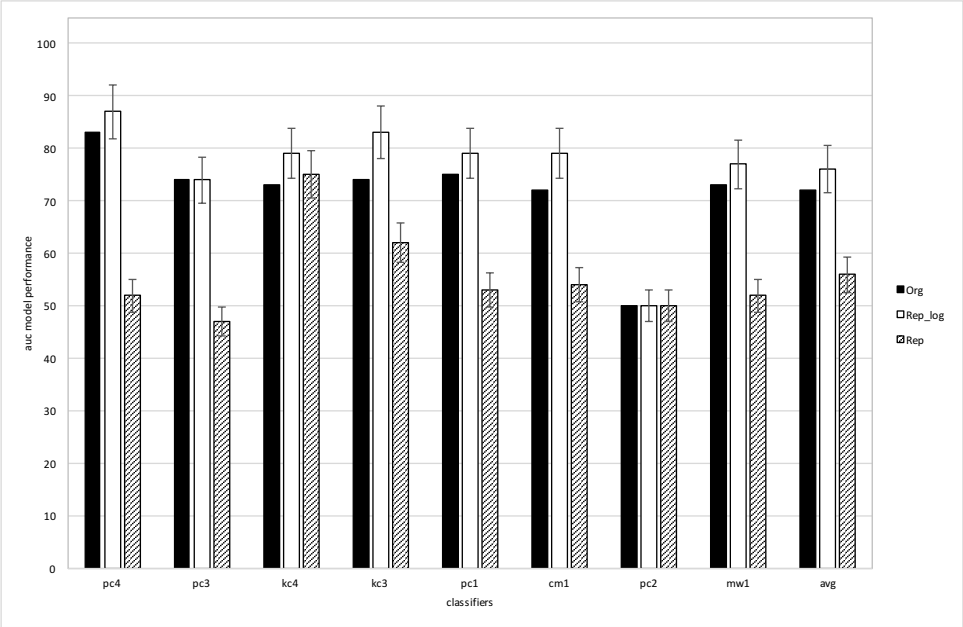
Results: Figure 6.2 shows the two runs that I performed for each classifier (SVM and VP). The percentage differences are given in Table 6.2. Figure 6.2a shows SVM's first run with a large difference between the original results in white and first reproduction in stripes. The result is completely not reproducible. Applying the log transform significantly increases SVM's performance in white, higher than the original results in black over 6 datasets (pc4, pc3, kc4, pc1, cm1, mw1) and the overall average. The same result is seen for one dataset (kc3), and lower performance for (pc2).

Figure 6.2b shows VP's results are close for (kc4) and exact for (pc2) with and without log transform. Among all datasets (pc2) is the second largest with over 4,000 modules with <1% defective modules. Which shows that VP is strongly affected by imbalanced data more than SVM on log transformed dataset. Subsequent results without log transform for VP are not reproducible. With the log transform (pc3) has an exact match while outperforming the original results on the remaining datasets (pc4, kc4, kc3, cm1, mw1) and the average (Table 6.2).

Overall my results in most cases are now aligned with the original results because of the log transform. The log transform is an important step and perhaps it is the missing preprocessing information not reported in the original study. These results provide opportunity to investigate further characteristics of the dataset family affecting results as mentioned in section 6.2 which is beyond this study. Having



(a) SVM



(b) VP

Figure 6.2: Bar charts showing two runs of Lessmann et al. [2008]'s study. The original's results are in black. The non-reproducible classifiers, Support Vector Machines (SVM) and Voted Perceptron (VP), are in stripes and white for first and second run. The first run in stripes is done without log transforming the dataset values. The second in white is done with log transform, this reduces the margin between small and large values of the features in the datasets, which significantly increases model performance for both classifiers.

discovered two potential factors that affect performance and reproducibility (data imbalance and data preprocessing) I investigate the final factor I consider to likely affect reproducibility in the next section.

6.4 Effect of Tool Version Difference Problem

As mentioned in the previous chapter, it was not possible to use the exact tools as all the original studies. In particular, Lessmann et al. [2008] used Matlab and Yale Workbench (now known as Rapid Miner) which were not used because I was familiar with the Java-based Weka. Lincke et al. [2008] reported that different tools give different results. My initial first year experiment was based on [Menzies et al. 2007] and [Rodriguez et al. 2014]. The Rodriguez et al. study compared different imbalance techniques. After reproducing the study, some differences were observed and I decided to check for every reproduction study which tool versions were used. Both [Menzies et al. 2007] and [Rodriguez et al. 2014] do not report the versions of Weka used. To find the right version of Weka, releases close to publication date of the original study were compared.

Data: 12 NASA datasets curated by Shepperd et al. [2013] were used.

Tool: WEKA was used in the original study. The version was not reported.

Experimental procedure of Rodriguez et al. [2014]: Several approaches that tackle data imbalance were compared. Using J48 as the base learner, random undersampling, synthetic minority oversampling technique (SMOTE), bootstrap aggregating (ensemble; building multiple learners and using majority voting for classification) were compared. Additionally, cost sensitive learners were used, by making cost of false negative to be 10 times more than false positive. Which means 1FN = 10FP, and importance is given to FN since it leads to failure of a software in operation. These classifiers were measured using MCC. A 5 by 5 cross validation was used for training and testing phase.

Results: Table 6.3 shows that the results are similar in the performances (MCC). The study provided

Table 6.3: Results of Rodriguez et al. [2014] reproduction study showing our actual results and the percentage difference in parenthesis. Negative values of % difference show that the replication performance is higher

		J48	RUS	ROS	SMOTE	SMOTEBoost	RUSBoost	MetaCost	CSC-Resamp	CSC-MinCost	AdaBoostM1	Bagging	RF
D' MCC	CM1	0.10 (0.00)	0.18 (0.39)	0.17 (0.35)	0.17 (0.35)	0.16 (-0.13)	0.16 (0.00)	0.23 (0.00)	0.23 (0.17)	0.13 (0.00)	0.19 (0.00)	0.12 (0.00)	0.04 (-0.25)
	JM1	0.23 (0.00)	0.24 (0.17)	0.23 (0.00)	0.24 (0.04)	0.26 (0.04)	0.24 (0.00)	0.25 (0.08)	0.24 (0.25)	0.21 (0.05)	0.24 (-0.04)	0.26 (-0.04)	0.18 (-0.11)
	KC1	0.28 (-0.04)	0.32 (0.13)	0.30 (0.03)	0.31 (0.06)	0.33 (-0.03)	0.34 (0.00)	0.33 (0.06)	0.32 (0.13)	0.21 (-0.19)	0.31 (0.00)	0.36 (0.06)	0.31 (0.10)
	KC3	0.22 (0.00)	0.25 (0.20)	0.24 (0.08)	0.29 (0.24)	0.29 (0.10)	0.26 (0.09)	0.22 (-0.09)	0.24 (0.33)	0.18 (0.00)	0.24 (0.00)	0.30 (0.00)	0.28 (0.32)
	MC1	0.44 (0.05)	0.20 (-0.80)	0.40 (0.48)	0.43 (0.51)	0.42 (-0.26)	0.17 (0.04)	0.35 (-0.20)	0.44 (0.18)	0.41 (0.12)	0.59 (0.00)	0.45 (-0.02)	0.45 (0.00)
	MC2	0.21 (0.00)	0.21 (0.00)	0.21 (0.00)	0.20 (-0.05)	0.34 (0.00)	0.36 (0.25)	0.16 (0.00)	0.16 (0.06)	0.18 (0.50)	0.32 (-0.22)	0.33 (0.21)	0.38 (0.32)
	MW1	0.32 (0.22)	0.22 (0.45)	0.10 (-2.10)	0.15 (-1.07)	0.19 (-0.21)	0.27 (0.37)	0.22 (0.32)	0.20 (0.10)	0.31 (0.06)	0.25 (0.00)	0.20 (-0.05)	0.30 (0.23)
	PC1	0.24 (0.04)	0.29 (0.03)	0.24 (-0.13)	0.26 (-0.04)	0.29 (-0.14)	0.33 (0.03)	0.29 (-0.07)	0.30 (0.10)	0.25 (0.08)	0.23 (-0.30)	0.25 (0.08)	0.22 (0.00)
	PC2	0.00 (0.00)	0.16 (0.25)	0.07 (-0.86)	0.09 (-0.44)	0.08 (0.13)	0.12 (0.42)	0.11 (0.36)	0.09 (-0.44)	0.00 (0.00)	0.01 (-2.00)	0.01 (0.00)	0.00 (0.00)
	PC3	0.24 (0.08)	0.25 (0.08)	0.22 (0.00)	0.22 (0.00)	0.30 (0.03)	0.31 (0.13)	0.32 (0.09)	0.29 (0.07)	0.29 (0.10)	0.29 (0.14)	0.23 (0.09)	0.19 (-0.05)
	PC4	0.51 (0.12)	0.52 (0.06)	0.47 (-0.04)	0.52 (0.06)	0.56 (0.04)	0.55 (0.02)	0.53 (0.02)	0.51 (0.08)	0.54 (0.06)	0.53 (0.00)	0.51 (0.02)	0.54 (0.11)
	PC5	0.50 (0.02)	0.52 (-0.02)	0.51 (-0.04)	0.54 (0.02)	0.55 (0.00)	0.48 (-0.15)	0.56 (0.07)	0.52 (-0.02)	0.52 (0.00)	0.52 (0.02)	0.52 (0.04)	0.52 (0.04)
	avg	0.27 (0.03)	0.28 (0.07)	0.26 (-0.04)	0.29 (0.07)	0.31 (-0.06)	0.30 (0.03)	0.30 (0.03)	0.29 (0.10)	0.27 (0.04)	0.31 (-0.03)	0.30 (0.07)	0.28 (0.07)

all artefacts (scripts, results in arff format, data set link). The challenges are missing parameters for performing the experiments in WEKA Filters for undersampling to handle imbalance data had errors at runtime. Some missing parameters in the paper were retrieved from the shared results. Even though these variations are small I compared multiple WEKA versions released around the publication date of the paper. Surprisingly, different versions of tools have different results as shown in Table 6.4.

Table 6.4: A reproduction study of Rodriguez et al. [2014] using many versions of Weka. Bagging is used for dealing with imbalance. The negative values of % difference show version 3-7-7 performs lower.

	Weka Versions		
Datasets	3-7-7	3-7-8 (to 10)	diff
CM1	0.07	0.12	**(-71)**
JM1	0.26	0.27	(-3)
KC1	0.34	0.34	(0)
KC3	0.32	0.30	**(6)**
MC1	0.45	0.46	(-2)
MC2	0.32	0.26	**(19)**
MW1	0.32	0.21	**(34)**
PC1	0.25	0.23	**(8)**
PC2	-0.01	0.01	**(200)**
PC3	0.21	0.21	(0)
PC4	0.53	0.50	(5)
PC5	0.52	0.50	(3)

Here Weka version 3.7.7 and 3.7.8 had 61 code parts removed while 58 were added. Additions and removals are denoted by plus and minus signs after running a 'diff' between the two files. The 'diff' command is used by a version control system (**git**) . It helps manage code and the different versions created throughout a development process. A hunk shows where both files differ, -493 is the line number in the Java file and 7 is number of lines afterwards, and vice versa.

I studied the Weka API and extracted the notable change. The Java code provided below (Figure 6.3) was the major addition to the later version. It allows the bagging algorithm create data partitions using the PartitionGenerator class. Instances are taken randomly from the main training set into each partition. The sampling is done with replacement, which means an instance taken in one round can still be taken

```
protected Random m_random;
   protected boolean[][] m_inBag;
@@ -493,7 +439,7 @@ public class Bagging
     // create the in-bag dataset
     if (m_CalcOutOfBag) {
       m_inBag[iteration] = new boolean[m_data.numInstances()];
-      bagData = resampleWithWeights(m_data, r, m_inBag[iteration]);
+      bagData = m_data.resampleWithWeights(r, m_inBag[iteration]);
     } else {
       bagData = m_data.resampleWithWeights(r);
       if (bagSize < m_data.numInstances()) {
@@ -672,12 +618,63 @@ public class Bagging
   }

   /**
+   * Builds the classifier to generate a partition.
+   */
+  public void generatePartition(Instances data) throws Exception {
+
+    if (m_Classifier instanceof PartitionGenerator)
+      buildClassifier(data);
+    else throw new Exception("Classifier: " + getClassifierSpec()
+                             + " cannot generate a partition");
+  }
+
+  /**
+   * Computes an array that indicates leaf membership
+   */
+  public double[] getMembershipValues(Instance inst) throws Exception {
+
+    if (m_Classifier instanceof PartitionGenerator) {
+      ArrayList<double[]> al = new ArrayList<double[]>();
+      int size = 0;
+      for (int i = 0; i < m_Classifiers.length; i++) {
+        double[] r = ((PartitionGenerator)m_Classifiers[i]).
+          getMembershipValues(inst);
+        size += r.length;
+        al.add(r);
+      }
+      double[] values = new double[size];
+      int pos = 0;
+      for (double[] v: al) {
+        System.arraycopy(v, 0, values, pos, v.length);
+        pos += v.length;
+      }
+      return values;
+    } else throw new Exception("Classifier: " + getClassifierSpec()
+                               + " cannot generate a partition");
+  }
+
+  /**
+   * Returns the number of elements in the partition.
+   */
+  public int numElements() throws Exception {
+
+    if (m_Classifier instanceof PartitionGenerator) {
+      int size = 0;
+      for (int i = 0; i < m_Classifiers.length; i++) {
+        size += ((PartitionGenerator)m_Classifiers[i]).numElements();
+      }
+      return size;
+    } else throw new Exception("Classifier: " + getClassifierSpec()
+                               + " cannot generate a partition");
+  }
```

Figure 6.3: Java code extracted from the Bagging algorithm of Weka version 3-7-7 and 3-7-8 by running a 'diff'. The code was added in the later version to create partitions of the date when building models.

again. A model is trained per partition, where an increase in number of partitions creates more trained models. Mean of all trained models are taken to get a generalised performance.

Consequently, this partitioning process is less prone to overfitting (i.e only performing well on training set). The results for Rodriguez et al. [2014] in Table 6.4 show that Bagging of version 3.7.7 (without partitioning code) has significantly higher results, a likelihood of overfitting.

Researchers need to exercise caution in terms of the versions of tools they use, especially when reporting experiments that they intend for others to reproduce. Missing version information can comprise the reproduction process due to implementation variations of algorithms within tools. Consequently, an original study may seem unreliable if reproduction results vary, whereas reporting 'just' the version can easily solve this problem.

6.5 Conclusion

The findings in this chapter contribute the answer to my (RQ5). Reproducibility checks are possible. However, four factors contribute to reproducibility failure when they are inconsistent between an original and its reproduction study. The four factors are data inconsistency, data imbalance, data preprocessing, and tool versions.

Overall controlling these factors is not sufficient to achieve complete reproduction. Most of the results are still not aligned with those of the originals. Keeping one version of a dataset and reporting of preprocessing steps and tool versions may reduce reproducibility failures and improve research verifiability. More reproduction studies and the investigation of factors affecting reproducibility need to be done. In the next chapter I discuss the implications of all my findings and provide a set of practical recommendations to encourage and improve reproduction and replication.

7. Conclusions

In conclusion this thesis is about the discovery of inherent threats to software defect prediction research through reproduction and replication. Scientifically, it would be beneficial for practitioners to reproduce and replicate all important results. So far, I supported my thesis by identifying that there are limited numbers of such studies, especially on high quality original papers. In my work, I have looked at all existing studies of this nature and analysed them: I have found examples of good practice in their methodologies, but also examples of bad practice. I have also attempted independent reproduction of five studies. Based on my analysis and practical work, I have uncovered some issues with existing studies and attempts. One outcome of this work is a set of practical steps to help support and encourage more and better quality reproductions and replications in the future.

In this chapter, I explain the limitations of my study. I then expand on the implications of my results. Such as the consequence of few replications in a field. I relate the 'few replications issue' in light of the recent concerns in defect prediction raised in chapter 1. The absence of consistent reporting and systematic replication is also discussed. I then provide practical recommendations to improve the state of replication in defect prediction. I summarise the answers to my research questions. Finally, I suggest important future work based on my contributions.

7.1 Limitations of this Research

My research is limited to a set of studies in defect prediction (not all studies). My work looks at the claims by Shepperd et al. [2014] that research may not be replicable due to bias of research groups. Shepperd et al. [2014] analysed studies from the 208 studies identified by Hall et al. [2012]. The Hall

et al. [2012] SLR provides the highest number of papers discovered in Software Engineering SLRs that I found. All the SLRs have the following number of studies: [Catal and Diri 2009] 74 papers, [Hall et al. 2012] 208 papers, [Radjenović et al. 2013] 106 papers, [Malhotra 2015] 64 papers, [Wahono 2015] 71 papers, and [Hosseini et al. 2017] 30 papers respectively. As such, I scope my analysis and reproduction approach to the most representative set of studies, 208 papers in defect prediction taken from the Hall et al. [2012] SLR.

The main threat (in chapter 4) is in the identification of papers that replicate from the 208 original studies and the search engine used (Google Scholar). The search ended in 2016, since then the replicated papers have been monitored automatically to trigger email alerts of any new papers that cite the original studies. The search string is saved and is run automatically by Google Scholar with every new citation of the replicated study. Each paper is checked to confirm if it was a replication or not; no new replication has been identified and I believe this threat has been mitigated.

There are different search engines (Scopus, ISI Web of Science etc.) and Google Scholar was chosen because it has been effective as demonstrated by Wohlin [2014] for this type of search. In addition between 2011 and 2012 Google Scholar has *"very significantly expanded its coverage... at a stable rate"* [Harzing 2014]. Primarily, I am concerned about getting a reliable number of citations for my analysis and not usability. Although I found it useful to reduce the number of papers to read manually due to the 'search within citing articles' feature. I am confident that Google Scholar is sufficient for my work.

Threats also exists in assessing and extracting information from the original studies and their accompanying replications. These threats were mitigated in the following way. Two authors (Mahmood and Bowes) in the chapter 4 study read and extracted information from 5 of the 39 papers and for the six factors extracted from all the 208 papers. Using the SLuRp tool [Bowes et al. 2012] any disagreements

were identified and then resolved and the data was updated accordingly. The statistical analysis of citation count and the number of replications involves data with many ties (since 8 of the replicated studies are replicated once and many are not replicated). The Kendall correlation analysis is not as accurate due to the ties.

Threats exist in reproducing experiments in chapter 5. These threats were mitigated by ensuring all experimental designs of the reproduction were validated with their original designs. In each experiment the validation was done by at least two people. In addition I put in place data consistency checks (chapter 5, chapter 6) for all datasets I downloaded from the data repositories given by the original studies. I recorded all dataset information reported in the original studies and the information I found from the datasets I downloaded from the repository. I then marked any inconsistencies and variations in the number of instances and proportion of the defective instances observed. This recorded information served as a reference point for continuous validation between the information I also reported in this dissertation and what I observed and uploaded in our online appendix; this is to ensure the experiments and data I shared is consistent with my reporting. Most threats have been minimised and believe to the best of my ability these findings are sound.

7.2 Implications of Findings

Overall my results show that defect prediction suffers from a lack of external replications with only 6% of 208 studies replicated. Silva et al. [2014] identified 96 articles, reporting 133 replications performed between 1994 and 2010 in software engineering. The result of their study indicates that replication in software engineering is carried out more frequently than in defect prediction. Having few replications in a field has a negative impact on its research and progress. Because practitioners must be skeptical about using the results and teachers must ignore the findings until supported by replication [Evanschitzky et al. 2007]. In the medical field, Chanock et al. [2007] reported that it is unlikely that 'one' study would have

a valid result without requiring replication. I believe few replications must not be accepted in scientific fields. Some of the replication studies I found have detected errors in highly influential original studies. The fewer the replications the more chances there are for influential studies to propagate erroneous results. Such erroneous papers can distort the very future foundation upon which new research will be based upon. My results strongly suggest the need for more replications. More replications should provide confidence to practitioners on the usefulness and reliability of defect prediction research.

It would be interesting to find out why there are few replications. Perhaps, few replication studies could be an indication for researchers to define new research goals. My results may simply demonstrate decreasing interest in defect prediction. Recent criticisms suggest the need for defect prediction to be more relevant to practitioners. Lanza et al. [2016] reported that the problem with defect prediction lies in how the approaches are evaluated and benchmarked. They further suggested that *"researchers should seriously consider putting their predictors out into the real world and having them used by developers who work on a live code"*[Lanza et al. 2016]. Shepperd [2017] mentioned that evaluation of these prediction models is problematic and *"that the concerns of researchers need to be better aligned with the likely end-users"* [Shepperd 2017]. Kitchenam [2017] talked in-depth about the 4Rs (Rigour, Reproducibility, Replication and Relevance) and how they are linked. With good Rigour, there is value in Replicating Reproducible work. It is to check their stability across multiple organisations provided they are relevant to what practitioners need. Kitchenham further mentioned that *"very few papers consider practical issues"*[Kitchenam 2017]. Issues such as cost to a bidding document, when software is suitable for the next release. And suggested the need to obtain more realistic datasets and collaborate more with industry partners. My results also support Kitchenham's suggestion. An important characteristic that leads to a paper being replicated is original studies being based on closed source industrial data. Indicating that researchers may achieve relevance and attract replications by collaborating with industry and obtaining realistic data.

One aim was to find out how consistent replications were with the original studies. My results show that of the 29 replication experiments I analysed, 18 (62%) results agreed with the original paper. Agreements indicate the original studies are valid and practitioners can have more confidence in them. The agreement rate is encouraging because it shows replications can succeed in defect prediction. It is worth noting that a successful replication depends on how sufficient an original study reported the data, tools and method they used. Conversely, 38% of the failed replications must not be neglected. Why these studies failed should provide lessons for researchers to learn from and produce replicable studies. Leslie Sage, one of the Senior Editors for Physical Science submissions to Nature, gave a talk on "How to publish a paper in Nature" [1]. Sage mentioned that if he cannot reproduce the results of a study, he will reject the paper. If a study cannot be reproduced or replicated it should be treated with caution. Further investigations need to be carried out to find the errors.

I made reproduction attempts on five studies to find exemplar studies that are replicable and reproduce. And to find out important factors that impact reproducibility of studies. What is particularly worrying is that four of the replicated studies I reproduced failed despite their replications reporting agreements. That is, my reproduction results are much greater than 5% difference compared to the original's results. Similarly, in most cases the results of the replications were also greater than 5% compared to the original. Replication agreement is based on the reported outcome of the statistical test used. What if the wrong statistic is used to confirm the initial finding? Madeyski and Kitchenham [2017] reported that previous software engineering results are confirmed by weak statistical tests. Most software engineers are not statisticians, and can easily apply the wrong statistics. I have also experienced this previously. For example, Ghotra et al. [2015] found statistical flaws in Lessmann et al. [2008]. The findings could be nullified. Expert statisticians must be consulted for the correct analysis method. So, it is important to conduct further work on measuring replication agreements - not only based on what is reported.

[1]This talk was given on the 6th April 2016 at the University of Hertfordshire, Room E351. This is a popular talk and also available on https://www.seti.org/seti-institute/weeky-lecture/how-publish-paper-nature.

Furthermore, reporting inconsistencies are problematic for interpreting the outcomes of replications. For example agreements are not always reported clearly. Performance values of original studies are not always reported in the replication. Variations in datasets sizes exists between datasets information reported on paper and the dataset downloaded from a repository. These reporting inconsistencies hinder comparison of results, synthesis of knowledge, and affect reproducibility outcomes.

One of the main goals of my analysis was to find features of studies that are likely to attract replication: venue, quality, and influence. Most of the replicated original studies do not satisfy $quality_{4_p}$ this despite being largely published in high impact venues. Such a potential lack of quality in original studies is surprising and suggests unreliable findings may be propagated. Researchers need to be aware of this and focus on building quality research. Methodological quality is overlooked and has the potential to amplify erroneous results. Data preprocessing, dealing with data imbalance, and tuning parameters are qualities of good practise. Publishing in top journals/conferences is also related to replication. The influence (number of citations) of a paper and of a journal/conference differ from the quality of a paper. Influence and place of publication should not be taken as determining factors for quality of a paper. For example, the MSR conference (ranked as satisfactory) has higher impact than the ESEM conference which is ranked as excellent - I discussed the rankings in chapter 4, see Table 4.9.

It is important to note that quality may change over time. Previous influential experiments in defect prediction measured model performance using Accuracy e.g. [Elish and Elish 2008]. Accuracy does not account for data imbalance when measuring model performance [Gray et al. 2009]. Many defect prediction studies use cross validation for their experiment. Cross validation may not be as useful for all defect prediction approaches [Shepperd 2017]. The idea is that the definition of quality may change over time based on new findings. It is imperative for researchers to adhere to quality. But quality should be reviewed frequently. The review could be by consensus of experts in the field to continuously help in producing more relevant and reliable studies.

It was important for my thesis to find out how replications are performed. There is no structured approach for conducting replications. Most of the replication studies I found make many changes to the original study at once. Making many changes at once is prone to the masking of important influential factors on prediction models. Replications need to be done much more systematically.

My results in chapter 4 show the important replication steps that have been missed out in each replication, based on the applied Gómez et al. [2014] replication taxonomy. Being systematic, first and foremost, requires a researcher to break down a study into separate components (Table 4.2). Where typical components of a defect prediction study include, tools, statistics, cross validation, feature selection, parameter optimisation etc. The experiment is then run with the same components (i.e. reproduced), if the components are open source - otherwise with similar components. Then an intentional change to the study components is done, one after the other while recording their effect on model performance. Systematic replication has the potential to uncover underlying factors affecting results. A good example of systematic replication is Song et al. [2011]. The study first reproduced the original study of Menzies et al. [2007] to confirm it, then performed several combinations of the components while recording the effect on model performance. The study concluded that different combinations of the important three factors - learning algorithm, data normalisation, and feature selection - for different datasets, are needed to build suitable models. Good examples of replications based on closed source original studies are Andersson and Runeson [2007] and Galinac Grbac et al. [2013].

I provide a detailed landscape of replication activities in defect prediction in chapter 4. Statistical analyses and the use of data mining tools are the most frequently changed in replications. Data cleaning and tuning model parameters are changed the least. More scrutiny, and questions should be raised. For example what assumptions are used for statistical tests? Why is data cleaning and tuning not considered? Which tools provide standard lines of code metrics? Data gathered in this way would help researchers and practitioners increase their learning and understanding on how to build, apply and analyse models.

I now provide some practical suggestions on some of my findings. The aim is to guide researchers on the important characteristics to include in their studies. Characteristics that could lead to replication, and consequently lead to the confirmation of their results to achieve consistency of findings. I suggest ways to improve the standards of existing replication and reproduction practises with a focal point on quality.

7.3 Practical Recommendations

[**Recommendation 1**] Highly cited papers should in particular be replicated as such papers tend to influence future defect prediction practice. Other papers should also be replicated.

[**Recommendation 2**] Use a replication infrastructure (e.g. OpenML [http://www.openml.org/] Vanschoren et al. [2013] or Zenodo [https://zenodo.org/]). Such infrastructures typically include an application programming interface API (Weka, R, REST, Java, .Net, Python, mlR, Moa) based repository. OpenML allows experiments to be configured on it and run on a user's machine. The repository keeps one version of datasets, the results, the protocol for easy sharing, and long-lived for researchers to use in future.

[**Recommendation 3**] Better use of existing reporting guidelines should be made. This requires the development of comprehensive software engineering reporting guidelines. These should be based on existing guidelines, including Runeson and Höst [2009] on case study design, Kitchenham et al. [2008] on empirical software engineering, Carver [2010] on reporting replications, Silva et al. [2014] on designing and reporting replication studies and Mende [2010] on replication remedies, pitfalls and challenges. Crucially, these guidelines must be collected and structured as a repository similar to the repositories that already exist in the Medical field (e.g. [Munafò et al. 2017] [http://www.equator-network.org]).

[**Recommendation 4**] Replication Impact Factors should be put into practice. As Schimmack says: *"Demonstrating replicability should become an important criterion of research excellence that can be used by funding agencies and other stakeholders to allocate resources to research that advances science"* [Schimmack 2016].

I suggest ways in which replication can be used for Impact Factors:

- Use number of replications per study as additional impact factor metric $R - index$ [Schimmack 2016]
- Use number of reproductions per study as additional impact factor metric $Repro_{index}$
- Use number of replications and reproductions as the most significant impact factor metric RR_{index}.

[**Recommendation 5**] Quality assessments (e.g. $quality_{4p}$) should be applied to original studies. Researchers should consider quality in two parts: the quality of the methodology and quality of the reporting. These quality checks should be made on original studies before replication to minimise the spread of potentially erroneous results.

[**Recommendation 6**] The replication of important studies needs to be incentivised. Currently there is little reason for a researcher to replicate a study, as original studies are more likely to be cited than a replication. Highly rated publication venues should specifically encourage replications.

[**Recommendation 7**] Reproduction should be carried out before replication. This will demonstrate how close the replicating authors can get to the original study. There is little point attempting to replicate results if reproduction is not possible because, e.g. the raw defect data is not both accessible and held in a secure source.

These suggestions are not exhaustive. I provide these in light of my empirical analysis of replication and reproduction of studies for researchers to make improvements or refute them.

7.4 Summary of answers to the Research Questions

The answers to the research questions asked in this dissertation are summarised in this section.

RQ1 What proportions of defect prediction studies are replicated?

Only 6% of 208 original studies were replicated, suggesting replication and reproducibility are largely neglected in defect prediction studies. The fewer the replications the more chances there are for studies to propagate erroneous results.

RQ2 What types of replications are performed?

Replication studies make many changes to original studies at once, whereas incremental changes on a component by component basis may highlight important factors that impact model performance.

RQ3 What features of a defect prediction study make it likely to be replicated?

My results suggest that highly cited studies based on industry closed source data published in the Transactions on Software Engineering journal (highest impact in software engineering) leads to a paper being replicated.

RQ4 Do original and replication studies in defect prediction agree?

Replications are likely to succeed and in some cases discover mistakes. A sizeable number also disagree. It is difficult to confirm agreements in published results as there is inconsistent reporting of the performance measures.

RQ5 What factors are likely to affect reproducibility in defect prediction studies?

Dataset information are not consistent between the data repository and original paper. The same dataset can have different proportions of defective files and number of instances. A change in the defective proportion (imbalance) could affect the performance of a model by about 18%. The same tool but different versions and data preprocessing also affect performance and if not reported may lead to reproducibility failures.

I hope that these findings drive discussions along the line of my suggestions. Attempting to reproduce or replicate studies should not be construed as a repressive endeavour. Rather, as scientific approaches to collaborate and stabilise findings and bring about new research questions. I encourage researchers to replicate and extend my results to get more insight into replication across Software Engineering.

7.5 Future Work based on Contributions to Knowledge

Based on the findings in this dissertation, here are some potential directions based on my contributions to knowledge, most of which have been stated in the practical recommendations section 7.3.

Tool Analyses: It would be valuable to do tool analyses based on my findings about tool inconsistencies between versions discovered in (chapter 6). The value in this is to standardise these tools and minimise contradictory results when different or even the same tools are used for reproducing and replicating experiments; lack of a standard for tools could compromise the validity of a study. The tool problem has been discussed by Lincke et al. [2008] and a good example of tool analyses is done by Marshall et al. [2014] on the best tool for conducting SLR. Based on how I extracted and structured the data from the set of papers I identified, several tools used across defect prediction surfaced, with a potential to conduct such a tool analysis (Table 4.4, Table A.2).

Applying my proposed methodology for utilising SLRs to identify a particular type of papers (in this case, replication papers): This new appraoch reduces waste of a set of studies discovered by SLRs (a tedious and time consuming work). The SLR papers using my methodology can be used to find papers of certain types, more replication studies on different dates. The SLR used in this dissertation covers original papers from 2000-2010 and their replications up to 2017. SLRs published by Wahono [2015] and Malhotra [2015] can be used as base sets for replicating my study by reapplying its methodology.

Build a SE Repository of Guidelines: This is detailed in section 7.3.

Replicating the remaining quality studies and correcting existing non quality studies: This is an important endeavour for researchers to take up. Only 3 out of the 36 quality original studies have been replicated; this is a unique opportunity for researchers to replicate these studies and help establish their validity, their stability across multiple datasets (a large amount of datasets have been newly released

by Shippey et al. [2016]). There is potential to create new benchmark algorithms, discover the most robust quality technique and perhaps leading to an extended quality criteria (enforceable by publication venues). The non quality studies, most of which failed based on poor quality data, can be re-validated on these new datasets.

THE ORIGINAL STUDIES

S. Amasaki, Y. Takagi, O. Mizuno, and T. Kikuno. A bayesian belief network for assessing the likelihood of fault content. In *Software Reliability Engineering, 2003. ISSRE 2003. 14th International Symposium on*, pages 215–226, Nov 2003. doi: 10.1109/ISSRE.2003.1251044.

C. Andersson and P. Runeson. A replicated quantitative analysis of fault distributions in complex software systems. *Software Engineering, IEEE Transactions on*, 33(5):273–286, May 2007. ISSN 0098-5589. doi: 10.1109/TSE.2007.1005.

M. D'Ambros, M. Lanza, and R. Robbes. An extensive comparison of bug prediction approaches. In *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*, pages 31–41, May 2010. doi: 10.1109/MSR.2010.5463279.

N.E. Fenton and N. Ohlsson. Quantitative analysis of faults and failures in a complex software system. *Software Engineering, IEEE Transactions on*, 26(8):797–814, Aug 2000. ISSN 0098-5589. doi: 10.1109/32.879815.

T.M. Khoshgoftaar and N. Seliya. Tree-based software quality estimation models for fault prediction. In *Software Metrics, 2002. Proceedings. Eighth IEEE Symposium on*, pages 203–214, 2002. doi: 10.1109/METRIC.2002.1011339.

Sunghun Kim, Thomas Zimmermann, E. James Whitehead Jr., and Andreas Zeller. Predicting faults from cached history. In *Proceedings of the 29th International Conference on Software Engineering*, ICSE '07, pages 489–498, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2828-7. doi: 10.1109/ICSE.2007.66. URL http://dx.doi.org/10.1109/ICSE.2007.66.

S. Lessmann, B. Baesens, C. Mues, and S. Pietsch. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *Software Engineering, IEEE Transactions on*, 34(4):485–496, July 2008. ISSN 0098-5589. doi: 10.1109/TSE.2008.35.

T. Menzies, J. Greenwald, and A. Frank. Data mining static code attributes to learn defect predictors. *Software Engineering, IEEE Transactions on*, 33(1):2–13, Jan 2007. ISSN 0098-5589. doi: 10.1109/TSE.2007.256941.

R. Moser, W. Pedrycz, and G. Succi. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In *Software Engineering, 2008. ICSE '08. ACM/IEEE 30th International Conference on*, pages 181–190, May 2008. doi: 10.1145/1368088.1368114.

T.J. Ostrand, E.J. Weyuker, and R.M. Bell. Predicting the location and number of faults in large software systems. *Software Engineering, IEEE Transactions on*, 31(4):340–355, April 2005. ISSN 0098-5589. doi: 10.1109/TSE.2005.49.

Adrian Schröter, Thomas Zimmermann, and Andreas Zeller. Predicting component failures at design time. In *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, pages 18–27. ACM, 2006.

Thomas Zimmermann and Nachiappan Nagappan. Predicting defects using network analysis on dependency graphs. In *Proceedings of the 30th International Conference on Software Engineering*, ICSE '08, pages 531–540, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-079-1. doi: 10.1145/1368088.1368161. URL http://doi.acm.org/10.1145/1368088.1368161.

Thomas Zimmermann, Rahul Premraj, and Andreas Zeller. Predicting defects for eclipse. In *Predictor Models in Software Engineering, 2007. PROMISE'07: ICSE Workshops 2007. International Workshop on*, pages 9–9. IEEE, 2007.

THE REPLICATION STUDIES

C. Andersson and P. Runeson. A replicated quantitative analysis of fault distributions in complex software systems. *Software Engineering, IEEE Transactions on*, 33(5):273–286, May 2007. ISSN 0098-5589. doi: 10.1109/TSE.2007.1005.

T. R. Devine, K. Goseva-Popstojanova, S. Krishnan, R. R. Lutz, and J. J. Li. An empirical study of pre-release software faults in an industrial product line. In *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, pages 181–190, April 2012. doi: 10.1109/ICST.2012.98.

Ekwa Duala-Ekoko and Martin P. Robillard. A detailed examination of the correlation between imports and failure-proneness of software components. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, ESEM '09, pages 34–43, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-1-4244-4842-5. doi: 10.1109/ESEM.2009.5316047. URL http://dx.doi.org/10.1109/ESEM.2009.5316047.

Tihana Galinac Grbac, Per Runeson, and Darko Huljenić. A second replicated quantitative analysis of fault distributions in complex software systems. *Software Engineering, IEEE Transactions on*, 39(4):462–476, April 2013. ISSN 0098-5589. doi: 10.1109/TSE.2012.46.

Baljinder Ghotra, Shane McIntosh, and Ahmed E Hassan. Revisiting the impact of classification techniques on the performance of defect prediction models. In *Proc. of the 37th Int'l Conf. on Software Engineering (ICSE)*, May 2015.

Maggie Hamill and Katerina Goseva-Popstojanova. Exploring fault types, detection activities, and failure severity in an evolving safety-critical software system. *Software Quality Journal*, 23(2):229–265,

June 2015a. ISSN 0963-9314. doi: 10.1007/s11219-014-9235-5. URL http://dx.doi.org/10.1007/s11219-014-9235-5.

Maggie Hamill and Katerina Goseva-Popstojanova. Exploring fault types, detection activities, and failure severity in an evolving safety-critical software system. *Software Quality Journal*, 23(2):229–265, June 2015b. ISSN 0963-9314. doi: 10.1007/s11219-014-9235-5. URL http://dx.doi.org/10.1007/s11219-014-9235-5.

Segla Kpodjedo, Filippo Ricca, Philippe Galinier, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. Design evolution metrics for defect prediction in object oriented systems. *Empirical Softw. Engg.*, 16 (1):141–175, February 2011. ISSN 1382-3256. doi: 10.1007/s10664-010-9151-7. URL http://dx.doi.org/10.1007/s10664-010-9151-7.

Sandeep Krishnan, Chris Strasburg, Robyn R. Lutz, Katerina Goseva-Popstojanova, and Karin S. Dorman. Predicting failure-proneness in an evolving software product line. *Information and Software Technology*, 55(8):1479 – 1495, 2013. ISSN 0950-5849. doi: http://dx.doi.org/10.1016/j.infsof.2012.11.008. URL http://www.sciencedirect.com/science/article/pii/S0950584912002340.

S. Lessmann, B. Baesens, C. Mues, and S. Pietsch. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *Software Engineering, IEEE Transactions on*, 34(4):485–496, July 2008. ISSN 0098-5589. doi: 10.1109/TSE.2008.35.

Marek Leszak. Software defect analysis of a multi-release telecommunications system. In *Product Focused Software Process Improvement*, pages 98–114. Springer, June 2005.

Paul Luo Li, Mary Shaw, Jim Herbsleb, P Santhanam, and Bonnie Ray. An empirical comparison of field defect modeling methods, 2005.

T. Mende and R. Koschke. Effort-aware defect prediction models. In *Software Maintenance and*

Reengineering (CSMR), 2010 14th European Conference on, pages 107–116, March 2010. doi: 10.1109/CSMR.2010.18.

Thilo Mende. Replication of defect prediction studies: Problems, pitfalls and recommendations. In *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*, PROMISE '10, pages 5:1–5:10, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0404-7. doi: 10.1145/1868328.1868336. URL http://doi.acm.org/10.1145/1868328.1868336.

T.H.D. Nguyen, B. Adams, and A.E. Hassan. Studying the impact of dependency network measures on software quality. In *Software Maintenance (ICSM), 2010 IEEE International Conference on*, pages 1–10, Sept 2010. doi: 10.1109/ICSM.2010.5609560.

Ahmet Okutan and Olcay Taner Yıldız. Software defect prediction using bayesian networks. *Empirical Software Engineering*, 19(1):154–181, 2014.

T.J. Ostrand, E.J. Weyuker, and R.M. Bell. Predicting the location and number of faults in large software systems. *Software Engineering, IEEE Transactions on*, 31(4):340–355, April 2005. ISSN 0098-5589. doi: 10.1109/TSE.2005.49.

R. Premraj and K. Herzig. Network versus code metrics to predict defects: A replication study. In *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*, pages 215–224, Sept 2011. doi: 10.1109/ESEM.2011.30.

Foyzur Rahman, Daryl Posnett, Abram Hindle, Earl Barr, and Premkumar Devanbu. Bugcache for inspections: Hit or miss? In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, ESEC/FSE '11, pages 322–331, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0443-6. doi: 10.1145/2025113.2025157. URL http://doi.acm.org/10.1145/2025113.2025157.

Pradeep Singh and Shrish Verma. An efficient software fault prediction model using cluster based classification. *International Journal of Applied Information Systems (IJAIS)*, 7(3):35–41, 2014.

Qinbao Song, Zihan Jia, M. Shepperd, Shi Ying, and Jin Liu. A general software defect-proneness prediction framework. *Software Engineering, IEEE Transactions on*, 37(3):356–370, May 2011. ISSN 0098-5589. doi: 10.1109/TSE.2010.90.

Ayşe Tosun, Burak Turhan, and Booktitle = Proceedings of the 5th International Conference on Predictor Models in Software Engineering Date-Added = 2015-08-18 08:39:26 +0000 Date-Modified = 2015-10-14 16:30:03 +0000 Doi = 10.1145/1540438.1540446 Isbn = 978-1-60558-634-2 Keywords = code metrics, defect prediction, network metrics, public datasets Location = Vancouver, British Columbia, Canada Numpages = 9 Pages = 5:1–5:9 Publisher = ACM Series = PROMISE '09 Title = Validation of Network Measures As Indicators of Defective Modules in Software Systems Url = http://doi.acm.org/10.1145/1540438.1540446 Year = 2009 Bdsk-Url-1 = http://doi.acm.org/10.1145/1540438.1540446 Bdsk-Url-2 = http://dx.doi.org/10.1145/1540438.1540446 Bener, Ayşe. New York, NY, USA.

B. Turhan and A. Bener. A multivariate analysis of static code attributes for defect prediction. In *Quality Software, 2007. QSIC '07. Seventh International Conference on*, pages 231–237, Oct 2007. doi: 10.1109/QSIC.2007.4385500.

Hongyu Zhang. On the distribution of software faults. *Software Engineering, IEEE Transactions on*, 34 (2):301–302, March 2008a. ISSN 0098-5589. doi: 10.1109/TSE.2007.70771.

Hongyu Zhang. On the distribution of software faults. *Software Engineering, IEEE Transactions on*, 34 (2):301–302, March 2008b. ISSN 0098-5589. doi: 10.1109/TSE.2007.70771.

Hongyu Zhang, Xiuzhen Zhang, and Ming Gu. Predicting defective software components from code

complexity measures. In *Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium on*, pages 93–96, Dec 2007. doi: 10.1109/PRDC.2007.28.

References

Ieee standard classification for software anomalies. *IEEE Std 1044-2009 (Revision of IEEE Std 1044-1993)*, pages 1–23, Jan 2010. doi: 10.1109/IEEESTD.2010.5399061.

Fumio Akiyama. An example of software system debugging. In *IFIP Congress (1)*, volume 71, pages 353–359, 1971.

Dag W Aksnes. Characteristics of highly cited papers. *Research Evaluation*, 12(3):159–170, 2003.

Francis Bacon. *Bacon's Essays: And Colours of Good and Evil*. Macmillan, 1890.

Albert L. Baker, James M. Bieman, Norman Fenton, David A. Gustafson, Austin Melton, and Robin Whitty. A philosophy for software measurement. *Journal of Systems and Software*, 12(3):277 – 281, 1990. ISSN 0164-1212. doi: https://doi.org/10.1016/0164-1212(90)90050-V. URL http://www.sciencedirect.com/science/article/pii/016412129090050V. Oregon Workshop on Software Metrics.

Monya Baker. 1,500 scientists lift the lid on reproducibility. *Nature*, 533(7604):452–454, May 2016. ISSN 0028-0836.

V. R. Basili, F. Shull, and F. Lanubile. Building knowledge through families of experiments. *IEEE Transactions on Software Engineering*, 25(4):456–473, Jul 1999. ISSN 0098-5589.

Victor R Basili and Richard W Selby. Comparing the effectiveness of software testing strategies. *IEEE transactions on software engineering*, (12):1278–1296, 1987.

Christian Bird, Adrian Bachmann, Eirik Aune, John Duffy, Abraham Bernstein, Vladimir Filkov, and Premkumar Devanbu. Fair and balanced?: Bias in bug-fix datasets. In *Proceedings of the the 7th*

Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering, ESEC/FSE '09, pages 121–130, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-001-2.

Gary D. Boetticher, Tim Menzies, Tom Ostrand, and Guenther H. Ruhe. 4th international workshop on predictor models in se (promise 2008). In *Companion of the 30th International Conference on Software Engineering*, ICSE Companion '08, pages 1061–1062, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-079-1.

David Bowes, Tracy Hall, and Sarah Beecham. Slurp: A tool to help large complex systematic literature reviews deliver valid and rigorous results. In *Proceedings of the 2Nd International Workshop on Evidential Assessment of Software Technologies*, EAST '12, pages 33–36, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1509-8.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001. ISSN 1573-0565.

Jeffrey C Carver. Towards reporting guidelines for experimental replications: A proposal. Citeseer, 2010.

Cagatay Catal and Banu Diri. A systematic review of software fault prediction studies. *Expert Systems with Applications*, 36(4):7346 – 7354, 2009. ISSN 0957-4174.

Stephen J Chanock, Teri Manolio, Michael Boehnke, Eric Boerwinkle, David J Hunter, Gilles Thomas, Joel N Hirschhorn, Goncalo Abecasis, David Altshuler, Joan E Bailey-Wilson, et al. Replicating genotype–phenotype associations. *Nature*, 447(7145):655–660, 2007.

Nitesh V. Chawla. C4.5 and imbalanced data sets: investigating the effect of sampling method, probabilistic estimate, and decision tree structure. In *In Proceedings of the ICML?03 Workshop on Class Imbalances*, 2003.

S.R. Chidamber and C.F. Kemerer. A metrics suite for object oriented design. *Software Engineering, IEEE Transactions on*, 20(6):476–493, Jun 1994. ISSN 0098-5589.

Marcus Ciolkowski. *An Approach for quantitative aggregation of evidence from controlled experiments in software engineering*. Fraunhofer-Verlag, 2011.

B. Curtis. Software metrics: Guest editor's introduction. *IEEE Transactions on Software Engineering*, SE-9(6):637–638, Nov 1983. ISSN 0098-5589. doi: 10.1109/TSE.1983.235270.

John Daly. *Replication and a multi-method approach to empirical software engineering research*. PhD thesis, University of Strathclyde, 1996.

Steven Davies, Marc Roper, and Murray Wood. Comparing text-based and dependence-based approaches for determining the origins of bugs. *Journal of Software: Evolution and Process*, 26(1): 107–139, 2014. ISSN 2047-7481.

Karim O. Elish and Mahmoud O. Elish. Predicting defect-prone software modules using support vector machines. *Journal of Systems and Software*, 81(5):649 – 660, 2008. ISSN 0164-1212. doi: https://doi.org/10.1016/j.jss.2007.07.040. URL http://www.sciencedirect.com/science/article/pii/S016412120700235X. Software Process and Product Measurement.

Timothy M Errington, Elizabeth Iorns, William Gunn, Fraser Elisabeth Tan, Joelle Lomax, and Brian A Nosek. An open investigation of the reproducibility of cancer biology research. *Elife*, 3:e04333, 2014.

Heiner Evanschitzky, Carsten Baumgarth, Raymond Hubbard, and J. Scott Armstrong. Replication research's disturbing trend. *Journal of Business Research*, 60(4):411 – 415, 2007. ISSN 0148-2963. doi: https://doi.org/10.1016/j.jbusres.2006.12.003. URL http://www.sciencedirect.com/science/article/pii/S0148296306002347.

N.E. Fenton and M. Neil. A critique of software defect prediction models. *Software Engineering, IEEE Transactions on*, 25(5):675–689, Sep 1999. ISSN 0098-5589.

Martin Fleischmann, Stanley Pons, and Marvin Hawkins. Electrochemically induced nuclear fusion of deuterium. *Journal of Electroanalytical Chemistry*, 261(2):301–308, 1989.

Antske Fokkens, Marieke van Erp, Marten Postma, Ted Pedersen, Piek Vossen, and Nuno Freire. Offspring from reproduction problems: What replication failure teaches us. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1691–1701, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

Davide Fucci, Giuseppe Scanniello, Simone Romano, Martin Shepperd, Boyce Sigweni, Fernando Uyaguari, Burak Turhan, Natalia Juristo, and Markku Oivo. An external replication on the effects of test-driven development using a multi-site blind analysis approach. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '16, pages 3:1–3:10, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4427-2.

Christopher Gandrud. *Reproducible research with R and R studio*. CRC Press, 2013.

Vahid Garousi and Jo???? ao M. Fernandes. Highly-cited papers in software engineering: The top-100. *Information and Software Technology*, 71:108 – 128, 2016. ISSN 0950-5849.

Omar S Gómez, Natalia Juristo, and Sira Vegas. Understanding replication of experiments in software engineering: A classification. *Information and Software Technology*, 56(8):1033–1048, 2014.

Jesús M. González-Barahona and Gregorio Robles. On the reproducibility of empirical software engineering studies based on data retrieved from development repositories. *Empirical Software Engineering*, 17(1):75–89, 2012. ISSN 1573-7616.

Steven N. Goodman, Daniele Fanelli, and John P. A. Ioannidis. What does research reproducibility mean? *Science Translational Medicine*, 8(341):341ps12–341ps12, 2016. ISSN 1946-6234.

D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson. Reflections on the nasa mdp data sets. *Software, IET*, 6(6):549–558, Dec 2012. ISSN 1751-8806.

David Gray, David Bowes, Neil Davey, Yi Sun, and Bruce Christianson. Using the support vector machine as a classification method for software defect prediction with static code metrics. In Dominic Palmer-Brown, Chrisina Draganova, Elias Pimenidis, and Haris Mouratidis, editors, *Engineering Applications of Neural Networks*, volume 43 of *Communications in Computer and Information Science*, pages 223–234. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-03968-3.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, #nov# 2009. ISSN 1931-0145.

T. Hall and D. Bowes. The state of machine learning methodology in software fault prediction. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 308–313, Dec 2012.

T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell. A systematic literature review on fault prediction performance in software engineering. *Software Engineering, IEEE Transactions on*, 38(6): 1276–1304, Nov 2012. ISSN 0098-5589.

Maurice H Halstead. *Elements of software science*. Elsevier computer science library : operational programming systems series. North-Holland, New York, NY, 1977.

David J Hand, Heikki Mannila, and Padhraic Smyth. *Principles of data mining*. 2001.

Anne-Wil Harzing. A longitudinal study of google scholar coverage between 2012 and 2013. *Scientometrics*, 98(1):565–575, 2014.

R Hosseini, B Turhan, and D Gunarathna. A systematic literature review and meta-analysis on cross project defect prediction. 2017.

Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Information retrieval in folksonomies: Search and ranking. Springer, 2006.

David Hovemeyer and William Pugh. Finding bugs is easy. *SIGPLAN Not.*, 39(12):92–106, December 2004. ISSN 0362-1340. doi: 10.1145/1052883.1052895. URL http://doi.acm.org/10.1145/1052883.1052895.

John PA Ioannidis. Why most published research findings are false. volume 2, page e124. Public Library of Science, 2005.

John PA Ioannidis. Acknowledging and overcoming nonreproducibility in basic and preclinical research. *Jama*, 317(10):1019–1020, 2017.

Andreas Jedlitschka. *An empirical model of software managers' information needs for software engineering technology selection: a framework to support experimentally-based software engineering technology selection.* University of Kaiserslautern, 2009.

Natalia Juristo, Sira Vegas, Martín Solari, Silvia Abrahão, and Isabel Ramos. A process for managing interaction between experimenters to get useful similar replications. *Information and Software Technology*, 55(2):215 – 225, 2013. ISSN 0950-5849. Special Section: Component-Based Software Engineering (CBSE), 2011.

Erik Kamsties and Christopher M Lott. An empirical evaluation of three defect-detection techniques. In *European Software Engineering Conference*, pages 362–383. Springer, 1995.

Maurice G Kendall. Rank correlation methods. 1962.

Sunghun Kim, Hongyu Zhang, Rongxin Wu, and Liang Gong. Dealing with noise in defect prediction. In *Software Engineering (ICSE), 2011 33rd International Conference on*, pages 481–490, May 2011.

Barbara Kitchenam. Back to basics - the 4r's of software estimation, 2017.

B. A. Kitchenham, E. Mendes, and G. H. Travassos. Cross versus within-company cost estimation studies: A systematic review. *IEEE Transactions on Software Engineering*, 33(5):316–329, May 2007. ISSN 0098-5589. doi: 10.1109/TSE.2007.1001.

Barbara Kitchenham. Procedure for undertaking systematic reviews. *Computer Science Depart-ment, Keele University (TRISE-0401) and National ICT Australia Ltd (0400011T. 1), Joint Technical Report*, 2004.

Barbara Kitchenham, Hiyam Al-Khilidar, Muhammed Ali Babar, Mike Berry, Karl Cox, Jacky Keung, Felicia Kurniawati, Mark Staples, He Zhang, and Liming Zhu. Evaluating guidelines for reporting empirical software engineering studies. *Empirical Software Engineering*, 13(1):97–121, 2008.

Barbara Kitchenham, O. Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. Systematic literature reviews in software engineering – a systematic literature review. *Information and Software Technology*, 51(1):7 – 15, 2009. ISSN 0950-5849. Special Section - Most Cited Articles in 2002 and Regular Research Papers.

Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Stanford, CA, 1995.

Thomas S Kuhn. *The structure of scientific revolutions*, volume 2. University of Chicago press Chicago, 1963.

Peter C. R. Lane, Daoud Clarke, and Paul Hender. On developing robust models for favourability analysis: Model choice, feature sets and imbalanced data. *Decision Support Systems*, 53(4):712–718, 2012.

Peter C.R. Lane and Fernand Gobet. Developing reproducible and comprehensible computational models. *Artificial Intelligence*, 144(1):251 – 263, 2003. ISSN 0004-3702.

Filippo Lanubile and Giuseppe Visaggio. Assessing defect detection methods for software requirements inspections through external replication.

M. Lanza, A. Mocci, and L. Ponzanelli. The tragedy of defect prediction, prince of empirical software engineering research. *IEEE Software*, 33(6):102–105, Nov 2016. ISSN 0740-7459.

Jeffrey T. Leek and Roger D. Peng. Opinion: Reproducible research can still be wrong: Adopting a prevention approach. *Proceedings of the National Academy of Sciences*, 112(6):1645–1646, 2015.

Meridith Levinson. Let's stop wasting $78 billion a year. *CIO*, 15(2):78–78, 2001.

Gernot A. Liebchen and Martin Shepperd. Data sets and data quality in software engineering. In *Proceedings of the 4th International Workshop on Predictor Models in Software Engineering*, PROMISE '08, pages 39–44, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-036-4.

Rüdiger Lincke, Jonas Lundberg, and Welf Löwe. Comparing software metrics tools. In *Proceedings of the 2008 International Symposium on Software Testing and Analysis*, ISSTA '08, pages 131–142, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-050-0.

G Lonchampt, L Bonnetain, and P Hieter. Reproduction of fleischmann and pons experiments. In *Sixth International Conference on Cold Fusion, Progress in New Hydrogen Energy*, page 113, 1996.

Lech Madeyski and Barbara Kitchenham. Would wider adoption of reproducible research be beneficial for empirical software engineering research? *Journal of Intelligent & Fuzzy Systems*, (Preprint):1–13, 2017.

Zaheed Mahmood, David Bowes, Peter C. R. Lane, and Tracy Hall. What is the impact of imbalance on software defect prediction performance? In *Proceedings of the 11th International Conference on Predictive Models and Data Analytics in Software Engineering*, PROMISE '15, pages 4:1–4:4, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3715-1.

Zaheed Mahmood, David Bowes, Tracy Hall, Peter C.R. Lane, and Jean PetriÄÂĞ. Reproducibility and replicability of software defect prediction studies. *Information and Software Technology*, 2018. ISSN 0950-5849. doi: https://doi.org/10.1016/j.infsof.2018.02.003. URL http://www.sciencedirect.com/science/article/pii/S0950584917304202.

Ruchika Malhotra. A systematic review of machine learning techniques for software fault prediction. *Applied Soft Computing*, 27(0):504 – 518, 2015. ISSN 1568-4946.

Jon Marangos. Faster than a speeding photon. *Nature*, 406(6793):243, 2000.

Christopher Marshall, Pearl Brereton, and Barbara Kitchenham. Tools to support systematic reviews in software engineering: A feature analysis. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14, pages 13:1–13:10, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2476-2.

T.J. McCabe. A complexity measure. *Software Engineering, IEEE Transactions on*, SE-2(4):308–320, Dec 1976. ISSN 0098-5589.

Michael CH McKubre. The importance of replication. In *14th International Conference on Cold Fusion*. Citeseer, 2008.

Tim Menzies, Bora Caglayan, Ekrem Kocaguneli, Joe Krall, Fayola Peters, and Burak Turhan. The promise repository of empirical software engineering data. *West Virginia University, Department of Computer Science*, 2012.

James Miller. Replicating software engineering experiments: a poisoned chalice or the holy grail. *Information and Software Technology*, 47(4):233 – 244, 2005. ISSN 0950-5849.

Henk F. Moed. Measuring contextual citation impact of scientific journals. *Journal of Informetrics*, 4 (3):265 – 277, 2010. ISSN 1751-1577.

D. Mugnai, A. Ranfagni, and R. Ruggeri. Observation of superluminal behaviors in wave propagation. *Phys. Rev. Lett.*, 84:4830–4833, May 2000. doi: 10.1103/PhysRevLett.84.4830. URL https://link.aps.org/doi/10.1103/PhysRevLett.84.4830.

Marcus R. Munafò, Brian A. Nosek, Dorothy V. M. Bishop, Katherine S. Button, Christopher D. Chambers, Nathalie Percie du Sert, Uri Simonsohn, Eric-Jan Wagenmakers, Jennifer J. Ware, and John P. A. Ioannidis. A manifesto for reproducible science. *Nature Human Behaviour*, 1:0021 EP –, 01 2017.

Thomas J. Ostrand and Elaine J. Weyuker. How to measure success of fault prediction models. In *Fourth International Workshop on Software Quality Assurance: In Conjunction with the 6th ESEC/FSE Joint Meeting*, SOQUA '07, pages 25–30, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-724-7.

P. Patil, R.D. Peng, and J.T. Leek. What should researchers expect when they replicate studies? a statistical view of replicability in psychological science. *Perspectives on Psychological Science*, 11 (4):539–544, 2016. cited By 7.

Jean Petrić, David Bowes, Tracy Hall, Bruce Christianson, and Nathan Baddoo. The jinx on the nasa software defect data sets. In *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*, EASE '16, pages 13:1–13:5, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3691-8.

Adam Porter and Lawrence Votta. Comparing detection methods for software requirements inspections: A replication using professional subjects. *Empirical software engineering*, 3(4):355–379, 1998.

Adam A Porter, Lawrence G Votta, and Victor R Basili. Comparing detection methods for software requirements inspections: A replicated experiment. *IEEE Transactions on software Engineering*, 21 (6):563–575, 1995.

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. ISBN 1-55860-238-0.

Danijel Radjenović, Marjan Heričko, Richard Torkar, and Aleš Živkovič. Software fault prediction metrics: A systematic literature review. *Information and Software Technology*, 55(8):1397–1418, 2013.

Irina Rish. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM, 2001.

FS Roberts. Measurement theory, encyclopedia of math, vol. 7, 1979.

Daniel Rodriguez, Israel Herraiz, Rachel Harrison, Javier Dolado, and José C. Riquelme. Preliminary comparison of techniques for dealing with imbalance in software defect prediction. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14, pages 43:1–43:10, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2476-2.

P. Runeson and A. Andrews. Detection or isolation of defects? an experimental comparison of unit testing and code inspection. In *14th International Symposium on Software Reliability Engineering, 2003. ISSRE 2003.*, pages 3–13, Nov 2003.

Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, 2009. ISSN 1382-3256.

Kristian Sandahl, Ola Blomkvist, Joachim Karlsson, Christian Krysander, Mikael Lindvall, and Niclas Ohlsson. An extended replication of an experiment for assessing methods for software requirements inspections. *Empirical Software Engineering*, 3(4):327–354, Dec 1998. ISSN 1573-7616. doi: 10.1023/A:1009724120285. URL https://doi.org/10.1023/A:1009724120285.

Ulrich Schimmack. The replicability-index: Quantifying statistical research integrity., Jan 2016. URL https://wordpress.com/post/replication-index.wordpress.com/920.

Carolyn B Seaman. Organizational issues in software development: An empirical study of communication. 1998.

Steven Shapin, Simon Schaffer, and Thomas Hobbes. *Leviathan and the air-pump*. Princeton University Press Princeton, 1985.

M. Shepperd, Qinbao Song, Zhongbin Sun, and C. Mair. Data quality: Some comments on the nasa software defect datasets. *Software Engineering, IEEE Transactions on*, 39(9):1208–1215, Sept 2013. ISSN 0098-5589.

M. Shepperd, D. Bowes, and T. Hall. Researcher bias: The use of machine learning in software defect prediction. *Software Engineering, IEEE Transactions on*, 40(6):603–616, June 2014. ISSN 0098-5589.

Martin Shepperd. Machine learning may be the answer but is it trustworthy?, 2017.

Thomas Shippey, Tracy Hall, Steve Counsell, and David Bowes. So you need more method level datasets for your software defect prediction?: VoilÀ! In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '16, pages 12:1–12:6, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4427-2.

Forrest J Shull and Victor R Basili. *Developing techniques for using software documents: a series of empirical studies*. PhD thesis, research directed by Dept. of Computer Science.University of Maryland, College Park, Md., 1998.

Fabio Q. Silva, Marcos Suassuna, A. César França, Alicia M. Grubb, Tatiana B. Gouveia, Cleviton V. Monteiro, and Igor Ebrahim Santos. Replication of empirical studies in software engineering research: A systematic mapping study. *Empirical Softw. Engg.*, 19(3):501–557, #jun# 2014. ISSN 1382-3256.

Jacek Śliwerski, Thomas Zimmermann, and Andreas Zeller. When do changes induce fixes? In *Proceedings of the 2005 International Workshop on Mining Software Repositories*, MSR '05, pages 1–5, New York, NY, USA, 2005. ACM. ISBN 1-59593-123-6.

M. Solari. Identifying experimental incidents in software engineering replications. In *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 213–222, Oct 2013. doi: 10.1109/ESEM.2013.26.

Gregory Tassey. The roles and economic impacts of technology infrastructure. *National Institute of Standards and Technology*, 2008.

Jason Van Hulse, Taghi M Khoshgoftaar, and Amri Napolitano. Experimental perspectives on learning from imbalanced data. In V. Ghahramani, editor, *Proceedings of the 24th international conference on Machine learning*, pages 935–942. ACM, 2007.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013.

Sira Vegas. What makes a good empirical software engineering thesis?: Some advice. 2015.

Romi Satria Wahono. A systematic literature review of software defect prediction: Research trends, datasets, methods and frameworks. *Journal of Software Engineering*, 1(1):1–16, 2015.

John B West. Robert boyle's landmark book of 1660 with the first experiments on rarified air. *Journal of Applied Physiology*, 98(1):31–39, 2005.

Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

Claes Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14, pages 38:1–38:10, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2476-2.

Murray Wood, Marc Roper, Andrew Brooks, and James Miller. Comparing and combining software defect detection techniques: A replicated empirical study. In *Proceedings of the 6th European SOFTWARE ENGINEERING Conference Held Jointly with the 5th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ESEC '97/FSE-5, pages 262–277, New York, NY, USA, 1997. Springer-Verlag New York, Inc. ISBN 3-540-63531-9.

Zhe Yu, Nicholas A. Kraft, and Tim Menzies. How to read less: Better machine assisted reading methods for systematic literature reviews. *CoRR*, abs/1612.03224, 2016.

A. Appendix

A.1 Publication: Reproducibility and replicability of software defect prediction studies

Information and Software Technology 99 (2018) 148–163

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Reproducibility and replicability of software defect prediction studies

Zaheed Mahmood*,[a], David Bowes[a], Tracy Hall[b], Peter C.R. Lane[a], Jean Petrić[a]

[a] University of Hertfordshire, UK
[b] Brunel University London, UK

ARTICLE INFO

Keywords:
Replication
Reproducibility
Software defect prediction

ABSTRACT

Context: Replications are an important part of scientific disciplines. Replications test the credibility of original studies and can separate true results from those that are unreliable.

Objective: In this paper we investigate the replication of defect prediction studies and identify the characteristics of replicated studies. We further assess how defect prediction replications are performed and the consistency of replication findings.

Method: Our analysis is based on tracking the replication of 208 defect prediction studies identified by a highly cited Systematic Literature Review (SLR) [1]. We identify how often each of these 208 studies has been replicated and determine the type of replication carried out. We identify quality, citation counts, publication venue, impact factor, and data availability from all 208 SLR defect prediction papers to see if any of these factors are associated with the frequency with which they are replicated.

Results: Only 13 (6%) of the 208 studies are replicated. Replication seems related to original papers appearing in the Transactions of Software Engineering (TSE) journal. The number of citations an original paper had was also an indicator of replications. In addition, studies conducted using closed source data seems to have more replications than those based on open source data. Where a paper has been replicated, 11 (38%) out of 29 studies revealed different results to the original study.

Conclusion: Very few defect prediction studies are replicated. The lack of replication means that it remains unclear how reliable defect prediction is. We provide practical steps for improving the state of replication.

1. Introduction

Defect prediction is a very active area of research in software engineering. However the quality of defect prediction modelling is regularly criticised [2,3]. Replications are an important way in which to identify the quality of original studies and to increase the confidence that we can have in results [4,5]. Replications also test the claim that "*most research findings are false*" [6] and that a "*little replication goes a long way*" to separate true research findings from false positives [7]. The more replication studies are performed, the more opportunities there are for defect prediction studies to be improved and the state-of-the-art to mature.

This paper aims to quantify the subsequent replications of 208 defect prediction studies identified by Hall et al. [1]. We use Wohlin's [8] forward snowballing approach to identify papers that cite these original 208 studies. Within these citing papers, we identify replications of the original 208 defect prediction studies. We compare the prediction performance of an original study with its accompanying replication study. We measure performance agreement between studies. Agreements or disagreements show replication success or failure, and also indicate the replicability of studies. We extract the characteristics of original studies which have been replicated. Knowing the characteristics of replicated original studies should help authors of primary studies produce studies more accessible to replication. We also present a landscape of how replications are done in defect prediction. We aim to answer the following four research questions:

RQ1: Are defect prediction studies replicated?
RQ2: How are replications performed in defect prediction?
RQ3: What features of a defect prediction study make it likely to be replicated?
RQ4: Do original and replication studies in defect prediction agree?

We make the following contributions. First, we present a methodology for analysing replications that is based on using an existing SLR. Second, we provide a small baseline set of 39 defect prediction studies

* Corresponding author.
E-mail addresses: z.mahmood4@herts.ac.uk (Z. Mahmood), d.h.bowes@herts.ac.uk (D. Bowes), tracy.hall@brunel.ac.uk (T. Hall), peter.lane@bcs.org.uk (P.C.R. Lane), j.petric@herts.ac.uk (J. Petrić).

https://doi.org/10.1016/j.infsof.2018.02.003
Received 29 May 2017; Received in revised form 6 February 2018; Accepted 7 February 2018
Available online 10 February 2018
0950-5849/

Table 1
The identified replications in this study that are mapped & tagged to the considered categories of the Gómez et al. [5] replication taxonomy.

Replication type	Protocol	Operationalisation	Populations	Experimenters	Replication name & changed (Δ) components
Literal	=	=	=	=	Repetition
Operational	=	=	=	≠	Δ-experimenter (A)
	=	=	≠	=	Δ-populations
	=	=	≠	≠	Δ-populations/-experimenter (B)
	=	≠	=	=	Δ-operationalisation
	=	≠	=	≠	Δ-operationalisation/-experimenters (C)
	=	≠	≠	=	Δ-operationalisation/-populations
	=	≠	≠	≠	Δ-operationalisation/-populations/-experimenters (D)
	≠	=	=	=	Δ-protocol
	≠	=	=	≠	Δ-protocol/-experimenters (E)
	≠	=	≠	=	Δ-protocol/-populations
	≠	=	≠	≠	Δ-protocol/-populations/-experimenters (F)
	≠	≠	=	=	Δ-protocol/-operationalisation
	≠	≠	=	≠	Δ-protocol/-operationalisation/-experimenters (G)
	≠	≠	≠	=	Δ-protocol/-operationalisation/-populations
	≠	≠	≠	≠	Δ-protocol/-operationalisation/-populations/-experimenters (H)
Conceptual	Unknown	Unknown	Unknown	Unknown	(only hypotheses are retained)

(originals with their corresponding studies) for researchers to use in future studies. Third, we identify a set of characteristics of original studies for researchers to incorporate into their work to encourage subsequent replication. Finally, we provide practical recommendations which could increase the number of replications performed.

The paper is structured as follows: Section 2 gives background about replication and related work. Section 3 details the methodology while Section 4 provides results. Threats to validity are given in Section 5 and the implications and recommendations for replication are discussed in Section 6. Finally, Section 7 concludes the study.

2. Background and related work

Defect prediction has *"many researchers continuously proposing novel approaches to predict defects in software systems"* [9]. Ioannidis [6] reports that there is a high risk of false results in rapidly growing fields with many research groups (defect prediction can be described this way). Moonesinghe et al. [7] shows that the probability of a research claim being true is increased by replications. Quantifying replications in defect prediction is therefore important.

The number of replications in software engineering has previously been investigated by da Silva et al. [4] who found 96 software engineering papers replicating 72 original software quality and testing studies between 1994 and 2010. A total of 70% of the replications were conducted after 2004, and 70% of those were self replications. Even though replication growth is evident, it does not keep pace with the growth of empirical primary studies; therefore, more external replications are needed [4]. We set out to quantify external replications in one strand of software engineering, i.e. defect prediction. We based our analysis on tracking the replications of a representative sample of defect prediction studies from Hall et al. [1] because the study is one of the *"very prominent 'gold sets' of published SLRs"* and the authors *"define their work in enough detail for us to construct data sets for simulations"* [10].

The terms replication and reproducibility are often interchanged, but they carry different meaning. Replication means to repeat an experiment by independent researchers within a different environment, with changes to the original study aimed at getting consistent results. Reproduction is to recompile the same artefacts used for a study, including data, analysis and procedures for validation [11,12] to get the same results.

Bias may be a major threat to repeatability. Shepperd et al. [13] found that bias introduced by researchers accounts for most of the variance in defect prediction-model performance. So,"*it matters more who does the work than what was done*" and "*Clearly Research Group is a basket for a number of concepts including prior knowledge, statistical and data processing skills, interests, opportunities, relationships with practitioners and so forth*" [13]. These bias factors suggest that a study done by a research group may not be repeatable by others. Previous work has looked at the reproducibility of data mining studies [14,15]. Defect prediction studies invariably are based on data mining. González-Barahona and Robles [14] propose a process model to gauge the reproducibility of data mining studies by identifying key elements of the research including: data source, retrieval methodology, raw dataset, extraction methodology, study parameters, processed dataset, analysis methodology, and results dataset.

Goodman et al. [16] suggest that in any scientific field the kind of replication must be clearly specified. We adopt part of the Gómez et al. [5] replication taxonomy that tracks changes made to components of an original study, and identifies the different types of replications that can be performed. The taxonomy was originally defined for software engineering human-centric experiments, but we adapt it to defect prediction experiments (Section 3 presents our adaption).

According to Gómez et al.'s [5] taxonomy, replication in software engineering can be categorised into three broad types (see Table 1). *Literal* is a type of replication done by authors of the original study. In effect, this type of replication is often named Repetition because no component of the original study is changed; the same experiment is run by the same authors using the exact tools on the same data to avoid bias in the results. Modifying any component of the original study changes the type of replication to *Operational*. For example, if different authors replicate an original study while data and tools remain the same, it is the *Operational* replication type with Changed-experimenter (in effect the same as reproduction). Under the Operational replication, 15 changes can be made to the original study, and each change is given the appropriate name to reflect the change (Table 1 identifies these changes) for example the populations being studied may change. The third replication type is called *Conceptual* because every aspect of the original study is changed except the hypotheses. Applying this taxonomy to new and existing replications is crucial in aggregating replication types and results, to consolidate and synthesise new knowledge.

We aim to identify the number of replicated original defect prediction studies, and identify characteristics of these studies likely to relate to a paper being replicated. The characteristics of the original study we focus on are: study quality, publication venue, citation count and dataset. We focus on quality because Aksnes [17] deems quality as the core knowledge that leads to further developments by other researchers, with lasting significance. We focus on publication venue and study influence as Garousi and Fernandes [18] report that highly cited papers make studies influential. Aksnes [17] also reports that such influential papers tend to be published in journals. We focus on dataset as the availability and usability of data is likely to influence replication

Table 2
Summarised $quality_{4p}$ criteria defined by Hall and Bowes [2][a,b] from [1].

$Quality_{4p}$ assessment phases	Details of phases
Phase 1: Establishing that the study is a prediction study.	-Is a prediction model reported? -Is the prediction model tested on unseen data?
Phase 2: Ensuring sufficient contextual information is reported.	-Is the source of data reported? -Is the maturity of data reported? -Is the size of data reported? -Is the application domain of data reported? -Is the programming language of the data reported?
Phase 3: Establishing that sufficient model building information is reported	-Are the independent and dependent variables clearly reported? -Is the granularity of the dependent variables reported? -Are the modelling techniques used reported?
Phase 4: Checking the model building data	-Is the fault data acquisition process described? -Is the independent variables data acquisition process described? -Is the faulty and non-faulty balance of data reported?

[a] Phase 1 assesses defect prediction methodological approaches.
[b] Phases 2, 3 and 4 assess reporting of prediction studies.

potential. Only the quality characteristics are not directly measurable. We use the $quality_{4p}$ assessment process to characterise the quality of original studies as used by Hall et al. [1]. The $quality_{4p}$ process assesses defect prediction studies in terms of whether they employ a reliable *methodological* approach to building prediction models and whether studies *report* sufficient information to comprehend a study [1] (Table 2 summarises the quality criteria). A more detailed description of $quality_{4p}$ is outlined in Hall and Bowes [2].

3. Methodology

Our methodology has six stages with each stage further broken down.

3.1. Stage 1: Identification of replication papers

We use as our base set of studies the 208 original studies published in the 2012 SLR in defect prediction [1]. We used forward snowballing [8] to identify papers that subsequently cite and replicate the 208 original studies between 2000–2017 (15th April). This means that we sift through papers that cite original studies, identifying all possible papers that may replicate an original SLR study.

We used Google Scholar to identify citing papers for each of the 208 original studies. On the *'cited by #papers'* page of each paper we used the ~Replicate OR ~Replication OR ~Replicated string and selected the *'search within citing articles'* feature. In effect, only papers that used these terms or their synonyms (denoted by tilde (~)) were returned. Applying this technique reduces the number of papers to be assessed as replications and reduces false positives. We then read from the returned results page, the paper title and its summarised phrases to identify if the paper was a replication of an original study in the 208. If not sufficient, we accessed the whole document to find the context in which the term was used, as suggested by Wohlin [8]. For this search the in-built search feature of the web browser used or document reader was used to find where the term replication is used. If the replication term is not in the document we read the paper in full to establish if it was a replication. Using this approach we identified a set of papers that replicated a subset of the 208 original studies.

3.2. Stage 2: Inclusion criteria

Our focus is to find external replications (i.e., replications not by the original authors, as these are considered true replications [4,19]) of the original studies. We exclude a paper if the replication is by any original author(s), or was extended work by any of the original author(s). If the author(s) have extended an original piece of work, we considered this work to be one paper, and any replication of either of these two is a replication of an original study. We consider any author (whether a lead author or not) to be an author of the paper. Consequently we track all replications by all authors of original studies.[1] We found 13 original studies that have been replicated by 26 replication papers. These 39 papers are our *final-set*.

3.3. Stage 3: Data extraction process

3.3.1. Tool for extraction

For reproducibility (i.e. the ability of our research to be compiled and produce the same result), we used our SLuRp tool.[2] SLuRp is a web-based tool developed to make Systematic Literature Reviews (SLR) reproducible and also provides effective information storage and retrieval. SLuRp was assessed as the best of the SLR tools by Marshall et al. [21]. We did not use all of SLuRp's functionality, many more useful SLR management features are described in Bowes et al. [20]. We provide the following steps as a summary of SLuRp together with how we used it for data extraction.

1. Import BIBTeX files and store references to all original and replicated studies.
2. Assign two researchers (authors of this paper) to independently store extracted information from each paper.
3. Allow researchers to modify and approve extracted information.
4. Disagreements between researchers are flagged by SLuRp.
5. Create forms based on contextual and methodological information that must be extracted from each paper.
6. Store extracted information in the SLuRp database.
7. Retrieve stored information using SQL queries and organise into result tables.
8. Export tables as LaTeX tables. Graphs and box plots are available.
9. Edit entire paper with SLuRp LaTeX editor, including results, tables and compiled to produce the final paper.

3.3.2. Extraction of selected data from final set

Three sets of data were extracted that allowed us to answer RQ2, RQ3 and RQ4. The first set of extracted data (for RQ2) characterises how defect prediction studies are performed. This dataset is based on the defect prediction characteristics presented in Hall et al. [1] and Hall and Bowes [2]. These characteristics include:

[1] To clarify: If paper P is authored by Anne, Ben and Ceri. And paper Q is authored by any of Anne, Ben or Ceri, paper Q is NOT in the set of replicated papers. If paper R is an extension of paper P and has none of the original authors, R is included.
[2] Bowes et al. [20] available at https://bugcatcher.stca.herts.ac.uk/bugcatchers/faces/slurp/SLuRp.xhtml.

1. dependent variables
2. independent variables
3. algorithms
4. dataset
5. tuning
6. cross validation
7. statistical analysis

This set of defect prediction characteristics data allows us to gain insights on replication practice and to categorise replications based on changes replications make to the original studies in terms of these characteristics. The information we collect allows us to categorise replications into their respective categories (as defined in Table 1). Of the 39 *final-set*of studies, 5 papers were read independently between two authors (by way of a validation check on the data extraction process) and their data extracted, while agreements were reached on this data extraction using SLuRp to minimise threats to validity. Information on the remaining papers was then extracted by one of the authors. The second set of data extracted (for RQ3) allows us to determine which features of defect prediction studies make it likely that an original study will be replicated. This set of data is: study quality, publication venue, citation count and dataset (as presented in Section 2). The extraction of this data is described in Section 3.5. The third set of data extracted (for RQ4) allows us to establish whether the results of a replication study are comparable to the original. Section 3.6 describes the process by which we establish study outcome agreement.

3.4. Stage 4: Categorisation of replications into types

The Gómez et al. [5] replication taxonomy (Table 1) requires understanding of a study and its individual components before being applied to categorise replications into types. We breakdown defect prediction study components for replication classification by adapting the general component structure proposed by Gómez et al. [5] (see Appendix A). These study components represent changeable aspects of an original study during its replication (as described above). Each component changed may assist in the discovery of unknown factors that affect replication results.

The component data extracted from the *final-set*of papers are organised into tables (see Appendix B). We mapped each replication study to its type in Gómez et al. [5] taxonomy based on changes researchers made to the original study components during replication. We detail the four components of a study as follows;

Protocol is the overall study design. In defect prediction the framework that pulls together different sub-components to build a prediction system is the overall study design (protocol). Table B.16 shows the protocol sub-components we have used are:

1. Cross validation scheme used
2. Whether parameter tuning was performed
3. Which statistics were used to compare performance results
4. Whether data cleaning was used

These factors are motivated by Hall et al. [1] and Hall and Bowes [2] as outlined previously in Section 3.3. The protocol is the design before it is implemented (i.e. operationalised).

Operationalisation has two aspects, cause and effect. The cause is the process of implementing the protocol and considers the implementation environment (as shown in Table B.17) we consider the following implementation factors (again motivated by Hall et al. [1] and Hall and Bowes [2]):

1. Tools used
2. Algorithms used
3. Independent variables used

It should be noted that algorithms have been embedded into data mining tools like Weka [22], in effect the tools carry out the treatments required to implement a prediction framework. Therefore such tools and their versions must be considered because they may cause differences in replication results. Effect is the process of determining and defining the aspects of a model to be measured and selecting the appropriate measure. Since measures already exist (e.g. recall; measures the proportion of actual defects a model correctly predicted), it is a question of which appropriately measures the effect of the treatments in the model's prediction outcome. Consequently Table B.17 shows that the final operationalisation factor we collect is the dependent variable.

Population is based on the systems analysed in studies. These systems are then mined from source code repositories (open or closed sources). Changing a repository to mine data also changes the population. Table B.18 shows that the population factors that are considered are:

1. Data source
2. Domain
3. Language
4. Granularity of defect data

The granularity, i.e. method or class level, where the defective or non-defective data are gathered is also part of this. The programming languages used, size of project (KLOC), maturity (years of use and development), etc. Changing any of these sub-components affects the population and likely the replication results.

Experimenters are the researchers that conducted the study.

3.5. Stage 5: Identification of factors associated with replication

For all 208 papers, as discussed previously, we extracted 6 factors to find out if any of the factors have a relationship with the number of subsequent replications:

- $quality_{4p}$
- Number of citations of a paper
- Publication venue
- Publication venue's impact factor
- Data sharing/availability

We extracted $quality_{4p}$ assessment outcomes using Hall et al.'s quality check for defect prediction studies [1] for the 208 original studies that have been replicated (see Table 2 for a summary of $quality_{4p}$).

$Quality_{4p}$ overlaps extensively with González-Barahona and Robles' [14] reproducibility criteria which includes checking the: data source, retrieval method, raw data, extraction method, study parameters, analysis method, results method, identification and description. Two elements of Barahona and Robles' [14] reproducibility criteria are missing in $quality_{4p}$ and these are data availability and data flexibility. We additionally collect availability data (i.e. an element's tendency to exist in the future). We explicitly checked all the links of each study to confirm if data are accessible (in September 2017). We additionally collect Barahona and Robles' [14] flexibility criteria, i.e. adaptability to different environments by extracting the formats of shared data in terms of e.g. csv, arff etc. For open or closed source code repositories, metrics (e.g. object oriented metrics calculated on defective/non-defective code) can be collected to form defect data used as input for building prediction models [Org1, 2, 3]. For example the NASA MDP program provided defect datasets calculated from the raw source code of critical systems (e.g. Flight and Satellite systems). The raw source code, being proprietary, were not available. However, it is possible to reproduce a study based on the defect data which was shared even though if it was generated from a closed source.

We extracted impact factor values for their publication venue from

Table 3
ERA ranking categories.

Rankings	Description
A*	Flagship conference, a leading venue in a discipline area
A	Excellent conference, and highly respected in a discipline area
B	Good conference, and well regarded in a discipline area
C	Other ranked conference venues that meet minimum standards
Unranked	A conference for which no ranking decision has been made

http://www.core.edu.au/conference-portal.

journalmetrics (details are in Table 9). We used the Source Normalised Impact Average (SNIPA) [23] values which are based on the average citation per paper of a journal in that subject area. In addition, we extracted the ratings of journal/conference venues from Excellence in Research for Australia (ERA). In 2009, the Australian Research Council consulted the public, expert reviewers and academic bodies to rank journals and conferences, and produced the ERA rankings. We used the ERA 2010 rankings since other ranking bodies only provide journal impact factors and omit any ranking of conferences. ERA has 5 ranks according to research quality, see Table 3.

3.6. Stage 6: Assessing agreements between studies

We checked whether the performance reported in the original studies matched those reported in the replications. If replications agree then original studies are replicable. We also assessed reproducibility (getting the same results) since replications tend to vary because of contextual differences. By comparing predictive performance measures for both original and replication studies in the same context (i.e. same data, classifiers, metrics etc.). If the performance is different by $< 1\%$ we assess this as having being *reproduced*. If the change is $< 5\%$, it is *similar* and if it is $> 5\%$, we classify this as *different*. We chose these values based on the intervals used in statistical testing, i.e. 1% probability and 5% probability using standard statistical tests.

4. Results

4.1. RQ1: Are defect prediction studies replicated?

> Only 6% of 208 original studies were replicated, suggesting replication and reproducibility are largely neglected in defect prediction studies.

Only 13 out of the set of 208 original studies were replicated by different researchers reported in 26 papers (Table 4): 6% of the original studies, a significantly lower rate than the 94% non-replicated original studies. Which means that the lack of replication is substantial, consequently, there is a significant number of studies that have not been confirmed to report valid results via replication.

4.2. RQ2: How are replications performed in defect prediction?

> Replication studies make many changes to original studies.

Overall Table 4 shows that all replication studies made changes to the original study. Typically replications made three sets of changes to components of original studies.

Two replication studies (Hamill and Goseva-Popstojanova (Rep[1,2]), Hongyu Zhang (Rep[3,4])) replicated more than one original study, these two papers appear twice making the number of replication studies 26; these papers then appear twice in the 'Replication studies' column of Table 4 as Hamill and Goseva-Popstojanova (Rep[1,2]), and Hongyu Zhang (Rep[3,4]).

Three original studies (Andersson and Runeson (Org[4]) (Rep[5]), Lessmann et al. (Org[5]) (Rep[6]), Ostrand et al. (Org[6]) (Rep[7])) also conducted replications of other original studies and within them, certain aspects of their study have also been replicated. For example Lessmann et al. (Rep[6]) replicated Menzies et al. (Org[7]) and built a new classifier benchmarking framework. Ghotra et al. (Rep[8]) subsequently replicated the new framework. Therefore Lessmann et al.would appear twice in the first two columns of Table 4 with (Org[5]) (Rep[6]).

Table 4 shows that many changes are made to studies:

Table 4
13 replicated original studies out of the 208 with their replication studies, data sets, replication types and agreements between studies.

Replicated original studies	Replication studies	Agreements	Operation-al rep.0[a]
D'Ambros et al. (Org[1])	Mende (Rep[9])	Yes, Yes	(A), (G)
Andersson and Runeson (Org[4])	Hamill and Goseva-Popstojanova (Rep[1])	Yes	(H)
	Zhang (Rep[3])	No	(H)
Lessmann et al. (Org[5])	Ghotra et al. (Rep[8])	Yes, No	(A), (H)
Ostrand et al. (Org[6])	Marek (Rep[10])	Partial	(H)
	Mende and Koschke (Rep[11])	Unknown	(G)
Fenton and Ohlsson (Org[8])	Andersson and Runeson (Rep[5])	Partial	(H)
	Grbac et al. (Rep[12])	Partial	(H)
	Ostrand et al. (Rep[7])	Yes	(H)
	Zhang (Rep[4])	No	(H)
	Devine et al. (Rep[13])	Yes	(H)
	Hamill and Goseva-Popstojanova (Rep[2])	No	(H)
Menzies et al. (Org[7])	Turhan and Bener (Rep[14])	Yes	(G)
	Zhang et al. (Rep[15])	Yes	(G)
	Lessmann et al. (Rep[6])	Yes	(G)
	Song et al. Rep[16]	Yes, No	(A),(H)
	Singh and Verma (Rep[17])	Yes	(H)
Moser et al. (Org[9])	Krishnan et al. (Rep[18])	Yes	(H)
Kim et al. (Org[10])	Rahman et al. (Rep[19])	Yes	(H)
Zimmermann and Nagappan (Org[2])	Tosun et al. (Rep[20])	Yes	(H)
	Nguyen et al. (Rep[21])	Yes	(H)
	Premraj and Herzig (Rep[22])	Yes	(H)
Amasaki et al. (Org[11])	Okutan and Yıldız (Rep[23])	Unknown	(H)
Schröter et al. (Org[12])	Duala-Ekoko and Robillard (Rep[24])	Yes	(H)
Zimmermann et al. (Org[3])	Kpodjedo et al. (Rep[25])	Unknown	(H)
Khoshgoftaar and Seliya (Org[13])	Li et al. (Rep[26])	Yes	(H)

[a] Replication name tags: (A) changed-experimenters, (G) changed-protocol/-operationalisation/-experimenters, (H) changed-protocol/-operationalisation/-populations/-experimenters.

Table 5
Study-components of original studies that were changed during replication.

Protocol	Stats	CrossVal	DataClean	Parameter tuning
	19	8	5	2
Operationalisation	IndepVar	DepVar	Algorithm	Tools
	15	4	12	14
Populations	Granularity	Domain	SourceCode	ProgLang
	14	12	11	7

Full field names: statistical analysis, cross validation, data cleaning, optimising parameters, independent and dependent variables, programming language.

1. Changed-experimenters (tag A, 3 papers)
2. Changed-protocol/-operationalisation/-experimenter (tag G, 5 papers)
3. Changed-protocol/-operationalisation/-populations/-experimenters (tag H, 21 papers)

Replications in which most components are changed together dominates. Table 4 shows that Mende, Song et al., Ghotra et al.replicate with sets of two study-component changes (A,G and A,H). With changed-experimenter (A) as the first change and (H) as the last, changes (B,..., F) have been omitted for all replications indicating gaps in steps that need to be taken during replications.

The data we synthesised from all studies (in Tables 11–13 for original studies, and in the appendix Tables B.16–B.18 for replication studies) depicts a landscape of some of the tools, algorithms, and statistical analyses used in defect prediction. Table 5 shows that the statistical test component has the most changes compared to parameter tuning with the least changes. Replications tend to focus more on finding the most suitable statistical methods to describe data (e.g. Zhang (Rep[3]) suggests distribution of software faults are better described as a Weibull distribution, not in terms of the Pareto principle as originally proposed by Fenton and Ohlsson (Org[8])). While tuning the parameters of the prediction models to improve performance is considered the least.

There are 3 replication studies (Rep[16]), (Rep[8]), (Rep[9]) that did multiple runs of a single original study. The first run reproduced the original study as it is, and the second run either modified the protocol (Rep[9]), (Org[1]) protocol by adding a cross validation step, Song et al. Rep[16] used feature selection that ensured the test instances are not seen by the prediction model), or dataset (Ghotra et al. (Rep[8]) used less noisy data and a new dataset, Song et al. Rep[16] also added more datasets). These multiple runs have implications for agreements between studies and the types of replications performed, though such multiple runs are generally good practice.

4.3. RQ3: What features of a defect prediction study make it likely to be replicated?

> Our results suggest that studies based on industry closed source data published in the Transactions on Software Engineering journal (highest impact in software engineering (during the time period covered by Hall et al. [1])) leads to a paper being replicated.

We analysed the factors we extracted from each paper statistically.[3] We use a χ^2 test to establish the relationship between each binary factor and replications and Kendall's Tau rank correlation to test the relationship between citations and replications (as citations is continuous data) (see Table 6).

Table 7 shows the data format of the papers with datasets. Table 7 shows that there are few papers using formats other than arff. The small numbers do not allow a sound statistical analysis to be carried out for the affect of flexibility on the ability to be replicated.

Table 6
The 208 papers categorised as having $quality_{4p}$, shared data, appeared in TSE w.r.t being replicated*.

	$Quality_{4p}$		Shared data		InTSE*	
Replicated	Yes	No	Yes	No	Yes	No
Yes	3	10	5	8	5	8
No	33	162	70	125	10	185

* chosen as TSE dominates in Table 8.

Table 7
The formats of the data and the number of papers which use the format and the availability of the data.

Data format (flexibility)	Not replicated	Replicated
arff	60	2
csv	0	1
csv, arff	2	1
csv, xml	1	0
excel	1	0
xml	3	1

Table 8
Statistical tests for assessing $quality_{4p}$, shared data, TSE and citations, individually against replications.

Chi Square test	χ^2	*p*-value		
$Quality_{4p}$* replication	0.322	0.570		
Shared data * replication	0.035	0.852		
InTSE * replication	20.237	< **0.0001**		
Kendall		z	τ	*p*-value
Citations * replication		4.7614	0.269	< **0.0001**

There were 85 venues in which the 208 papers appeared (Online-Appendix [4]). Only 6 venues published papers that were subsequently replicated: PROMISE, MSR, ESEM, ISSRE, ICSE and TSE. TSE has the highest number of papers published with subsequent replications (Table 9). Table 8 shows that papers published in TSE are more likely to be replicated. We do not consider the impact factor of venues directly since, for non-replicated studies, impact factors are not available for many (63) publication venues.

Table 8 shows that a paper's influence (citations) has an impact on replication. However the quality of original papers or shared data use is not associated with subsequent replication.

Table 9 shows 10 of the 13 replicated studies have not passed the $quality_{4p}$ assessments. A replication not based on $quality_{4p}$ has ramifications on the validity of findings. For instance, data cleaning of the $quality_{4p}$ may have been overlooked or not reported, an indication that some findings may be erroneous. It is particularly true for the noisy NASA datasets used by 59 original studies (Table OA.1 in Online-Appendix).

Table 10 shows that 21 of 26 replication studies replicated original studies which were based on closed source industrial data (these will have needed to be replicated with different datasets). This suggests that studies based on closed source industrial data may be more attractive for replication.

[3] Using `R 3.3.1` open source statistical software.

[4] https://bugcatcher.herts.ac.uk/replication/Online-Appendix.html.

Table 9
The replicated original studies and whether they extracted contextual factors.

Citations	Original studies	Journal	ERA[a]	Impact[b]	$Quality_{4p}$ assessment[c] failed at phase	#Reps
128	Andersson and Runeson (Org[4])	TSE	A*	4.423	Phase1 No prediction done	2
577	Lessmann et al. (Org[5])	TSE	A*	4.423	Phase2 NASA data used	1
535	Ostrand et al. (Org[6])	TSE	A*	4.423	Phase4 Model building	2
684	Fenton and Ohlsson (Org[8])	TSE	A*	4.423	Phase1 No prediction done	6
816	Menzies et al. (Org[7])	TSE	A*	4.423	Phase2 NASA data used	5
361	Moser et al. (Org[9])	ICSE	A	2.988	Pass all	1
330	Kim et al. (Org[10])	ICSE	A	2.988	Phase4 model building	1
393	Zimmermann and Nagappan (Org[2])	ICSE	A	2.988	Phase4 model building	3
240	D'Ambros et al. (Org[1])	MSR	C	1.876	Pass all	1
43	Amasaki et al. (Org[11])	ISSRE	A	1.383	Phase2 contextual information	1
159	Schröter et al. (Org[12])	ESEM[d]	A	0.992	Pass all	1
126	Khoshgoftaar and Seliya (Org[13])	ESEM[e]	A	0.992	Phase2 contextual information	1
508	Zimmermann et al. (Org[3])	PROMISE	U	0.001	Phase2 contextual information	1

[a] CORE contributed to ERA rankings. Both rankings agree except on ICSE; A by ERA, A* by CORE (TSE not ranked).
[b] Source is journalmetrics, 2015 source normalised impact (SNIPA); takes average citation per paper of a journal in subject area.
[c] QA details summarised in Table 2.
[d] ISESE,
[e] METRICS are now part of ESEM http://www.esem-conferences.org/history.php .

Table 10
Descriptions of replicated original studies based on the type of data source and defect data sharing.

Source code	Shared data	No. of papers	No. of reps
Closed	No	6	15
	Yes	2	6
Open	No	2	2
	Yes	3	3

originals (i.e. where some of the replicated results were the same as the originals but not all). Additionally 3 studies did not report the level of agreement with the original study.

Our results show a variety of disagreements between the original and replicated results. There are a range of reasons for these disagreements that we will now discuss. Song et al. Rep[16] did 2 replication runs of Menzies et al. (Org[7]). In the first run the replication agrees with Menzies et al. (Org[7]). In the second run, Song et al. Rep[16] disagreed and report a flaw in (Org[7])'s attribute selection approach which meant that the test data included seen information and

Table 11
Protocol: Original studies.

Org. Studies	Cross Val.	Parameter tuning	Statistics	Data Cleaning
Andersson and Runeson (Org[4]), (Rep[5])	No	No	Pearson product-moment correlation	Yes: Duplicate failures
Lessmann et al. (Org[5]), (Rep[6])	Hold-out set	Yes	Friedmans test (rank classifiers), Nemenyi post hoc (statistical significance test on classifiers)	No
Ostrand et al. (Org[6]), (Rep[7])	No	No	*t*-test	No
Fenton and Ohlsson (Org[8])	No	No	Alberg diagrams	No
Menzies et al. (Org[7])	10 by 10 randomised	No	Quartile chart	No
Moser et al. (Org[9])	10 by 10 randomised	No	Kruskal–Wallis test	No
Kim et al. (Org[10])	No	No	No	No
Zimmermann and Nagappan (Org[2])	split-sample	No	Spearman correlation, Pearson, Nagelkerke (predictive power of logistic regression models), F-tests	No
Amasaki et al. (Org[11])	No	No	Error rate, Fishers exact test (correlation between 2 variables)	No
Schröter et al. (Org[12])	random splits	No	Two *t*-test, Spearman rank correlation	No
D'Ambros et al. (Org[1])	50 by 10fold randomised	No	*F*-test (explanative significance), Spearman correlation (evaluating predictive power of models) with Spearman coefficient (skewed data)	No
Zimmermann et al. (Org[3])	No	No	Spearman correlation, Pearson correlation	No
Khoshgoftaar and Seliya (Org[13])	10 fold cross validation	No	*Z*-test	No

4.4. RQ4: Do original and replication studies in defect prediction agree?

> It is difficult to confirm agreements in published results as there is inconsistent reporting of the performance measures.

Overall our analysis shows that the performance of 18 replicated experiments.[5] agreed with original performance values. This suggests that 62% of the replicated experiments were successful. The performance of 5 replicated experiments (17%) did not agree with originals and 3 replicated experiments (10%) resulted in partial agreement with therefore inflated performance of the defect prediction models.

Ghotra et al. (Rep[8]) did 2 replication runs of Lessmann et al. (Org[5]). The first run was based on uncleaned NASA data (including duplicate and inconsistent instances, see [24]) to confirm if no single classifier is best as in the original (Org[5]). The Friedman test used in Lessmann et al. (Org[5]) showed the ranking of model performances are not random; subsequently Nemenyi post hoc test was applied to detect which of the classifiers differed significantly. Ghotra et al. (Rep[8]) agree with Lessmann et al. (Org[5]) in the first run with the same data and different statistics, but disagree in the second run with a cleaned dataset curated by Shepperd et al. [25] and different statistics. In the second run, Ghotra et al. (Rep[8]) reported;

[5] Some papers conduct more than one experiment, there are 26 papers running 29 experiments.

Table 12
Operationalisation: Original studies.

Org. Studies	Tools	Algorithms	Independent Var	Dependent Var
Andersson and Runeson (Org[4]), (Rep[5])	No	No	Size (LOC)	Number of faults, fault density pre-release and post-release
Lessmann et al. (Org[5]), (Rep[6])	YALE machine learning	Statistical(7), Nearest Neighbours(2), Neural Networks(3), SVMs(5), Tree-based (3), Ensembles (2)	static code metrics	Defective or Not-defective
Ostrand et al. (Org[6]), (Rep[7])	VCS	Negative binomial regression	code age, programming language, logKloc,file status, release	Number of faults
Fenton and Ohlsson (Org [8])	ERIMET (metrics),FCTOOL (formal description language)	No	Complexity (McCabe cyclomatic complexity), size (LOC), communication (SigFF;new and modified signals count, inter and intra modules)	Number of faults, fault density prerelease and post-release
Menzies et al. (Org[7])	WEKA	OneR, J48, and Naive Bayes	Static code metrics	Defective or Not defective
Moser et al. (Org[9])	WEKA	Logistic regression, Naive Bayes, J48 (version 8)	Process, change, static code	Defective or Not defective
Kim et al. (Org[10])	Kenyon Infrastructure, APFEL (metrics), SVN, CVS	Least recently used (LRU)	spatial locality, temporal locality,Changed-entity and new-entity locality (churn)	Number of faults
Zimmermann and Nagappan (Org[2])	MaX (dependency information tracker), Ucinet 6 (network metrics)	Linear and logistic regression	Network measures on Dependency graphs, OO metrics, static code metrics	Number of defects, Defective or Not defective
Amasaki et al. (Org[11])	Netica (bayesian belief network software)	Bayesian Belief Network	design (product size), effort (person-day), detected faults, test items	Number of residual faults at acceptance test
Schröter et al. (Org[12])	R, BUGZILLA, CVS, SZZ	Linear and Ridge regression, Regression trees, SVM	import relationships (e.g. org.eclipse.ui) packages and imported classes	Number of defects, Defective or Not defective (post-release)
D'Ambros et al. (Org[1])	CVS, SVN, Bugzilla, Jira, Famix-Compliant OO model (scm metrics), Infusion (source code converter into FAMIX model), Moose (scm calculator), Churasco (history model, bug data extractor, classes linker, system files and bugs versioner)	linearReg	process, change, entropy of change, entropy, churn of source code, source code metrics, CK, OO	Number of defects (post-release)
Zimmermann et al. (Org [3])	Java parser (complexity metrics)	Linear regression (ranking), logistic regression (classification)	static code metrics (complexity), structure of abstract syntax tree (no of nodes etc.)	Number of defects, Defective or Not defective (prerelease, post-release)
Khoshgoftaar and Seliya (Org[13])	S-Plus (advanced data analysis), EMERALD - Datrix(fault data collection)	Least squares tree, s-plus, least absolute deviation	Design	Number of faults

Table 13
Populations: Original studies.

Org. Studies	Prog. Lang.	Domain	Granularity	Source	Availability
Andersson and Runeson (Org[4]), (Rep[5])	C, Java	Telecom	Module	Commercial	Not shared
Lessmann et al. (Org[5]), (Rep[6])	C, Java	Satellite, Flight, Storage	Module (predictions), code (metrics)	NASA, PROMISE	Shared
Ostrand et al. (Org[6]), (Rep[7])	Java, C, Makefiles, sql, shell, html, other	Inventory System, Provisioning System	File (predictions)	Industrial	Not Shared
Fenton and Ohlsson (Org[8])	No	Telecom	Module	Ericsson Telecom AB	Not shared
Menzies et al. (Org[7])	C, Java	Satellite, Flight, Storage	Method	NASA, PROMISE	Shared
Moser et al. (Org[9])	Java	IDE	File (predictions)	Eclipse 2.0, 2.1, 3.0	Not shared
Kim et al. (Org[10])	C, C++, Java	Web server, Browser, Text editor, Version control, Database, IDE, Email client	File, method (predictions)	Apache 1.3, JEdit, Subversion, PostgreSQL, Columba, Eclipse, and Mozilla	Not shared
Zimmermann and Nagappan (Org[2])	C++	Operating System	Binaries (predictions), Binaries (metrics)	Windows Server 2003	Not shared
Amasaki et al. (Org[11])	No	Embedded software	development process (metrics), directed graphs	Industrial	Not shared
Schröter et al. (Org[12])	Java	IDE	File, package (predictions), import packages and classes (metrics)	ECLIPSE plug-ins 52nos	Shared
D'Ambros et al. (Org[1])	Java	IDE	Class (predictions, metrics)	Eclipse (Mylyn, Equinox, PDE, Lucene, Score)	Shared
Zimmermann et al. (Org[3])	Java	IDE	Files, packages (predictions, metrics)	Eclipse 2.0, 2.1, 3.0	Shared
Khoshgoftaar and Seliya (Org[13])	Protel	Telecom	Modules (predictions), design documents (metrics)	Industrial	Not shared

Agree: replications that confirm original results, Disagree: replications that do not confirm original results, Partial: replications that confirm part of the original results, Unknown: replication that does not report agreement or disagreement.

> "*We used the Scott–Knott test to overcome the confounding issue of overlapping groups that are produced by several other post hoc tests, such as Nemenyis test [13], which was used by the original study. Nemenyis test produces overlapping groups of classification techniques, implying that there exists no statistically significant difference among the defect prediction models trained using many different classification techniques.*"

The curated data by Shepperd et al. [25] has been cleaned further by Petrić et al. [24]. The data errors found during this further cleaning may have also affected previous models. Overall, these findings suggest that replication leads to the discovery of mistakes and provides the opportunity to remedy those shortcomings.

Overall our results suggest that replications in defect prediction are possible with or without $quality_{4p}$ in the original study. Of the 29 studies, partial agreements (3) and disagreements (5) make up to 28% of the results, indicating that replication is able to detect errors and limitations of studies. Unreported (3) replications results (10%) are relatively high. We suggest that all replications need to state agreements and disagreements.

For a more detailed assessment of agreements, we extracted the performance values of replications with only 'changed - experimenter', as this type of replication is useful for assessing the reproducibility of research. Reproducibility aims to get the same result as the original study [11,12]. We categorise a paper as reproducible if the difference in the performance between an original and its replication does not go beyond 5%. We identified 5 replications of Menzies et al. ((Org[7]) shown in Table 14). Table 14 shows that Turhan and Bener (Rep[14]), and Zhang et al. (Rep[15]) report $>5\%$ different recall performance (64% and 85%), which means that neither study has succeeded in reproducing Menzies et al. (Org[7]) (71% recall). Table 14 also shows that Lessmann et al.'s replication (Rep[6]) used a different measure (auc) to the original measure reported making it difficult to assess reproducibility; similarly Song et al. Rep[16] reported only one performance measure (balance).[6] These results show that reporting inconsistencies between replications and original studies make it difficult to confirm agreements.

We investigate reproducibility further by ourselves reproducing (Org[5]), (Org[7]). Table 14 shows that our results are mixed despite matching closely all study components. In reproducing these original studies a number of anomalies with the original studies arose which may explain the differences in our performance values compared to the original studies. These anomalies include that the datasets we downloaded varied from the original in terms of number of defective units, number of instances etc. and also that our feature selection outcomes were not the same as the originals. Our Online-Appendix provides full details of these anomalies.

5. Threats to validity

The main threat to validity is that replication is currently performed so seldom that it is difficult to draw conclusions from the population of replications that we have. Many more replications need to be performed before it is possible to draw highly reliable conclusions about replication.

Another important threat is the identification of papers that replicate the 208 original studies and the tool used for the search, that is Google Scholar. The main search ended in 2016 and since then we have automatically monitored replicated papers with triggered mail alerts of new citing papers. The search string is saved and is run automatically by Google Scholar with every new citation of the replicated study. Each paper is checked to confirm if it was a replication or not; no new

[6] Mende (Rep[9]) reproduced D'ambros et al. (Org[1]), the results are mostly the same about ($< 1\%$), but with a few differences that could not be explained. We do not include Mende (Rep[9]) due to lack of space, but all the results can be found in our replication package https://bugcatcher.herts.ac.uk/replication.

Table 14
Model performance measure from Menzies et al. set of replications and Lessmann et al.

Data	Naive Bayes							SVM			VP		
	Recall					auc	balance	auc					
	(Org[7])	Us	(Rep[14])	(Rep[15])	(Rep[17])	(Rep[6])	(Rep[16])	(Org[5])	Us^{log}	Us	(Org[5])	Us^{log}	Us
pc4	98	87	–	–	72.6	85	82.6	92	95	50	83	87	52
pc3	80	79	–	–	80.6	81	71.4	77	94	53	74	74	47
kc4	79	80	–	–	–	68	71.9	77	85	60	73	79	75
kc3	69	78	–	–	99	83	74.1	86	85	50	74	83	62
pc1	48	73	–	–	66.2	79	64.6	80	94	56	75	79	53
cm1	71	77	–	–	81.5	72	72.7	70	96	51	72	79	54
pc2	72	86	–	–	83.3	85	81.8	85	71	50	50	50	50
mw1	52	78	–	–	100	80	70.5	65	83	50	73	77	52
avg	71	79.7	64	85	83	79	74	79	89	53	72	76	56

NB: Us^{log} denotes our results with a log transformation.

replication has been identified and we believe this threat has been mitigated.

There are different search engines (Scopus, ISI Web of Science etc.) and we chose Google Scholar because it has been effective as demonstrated by Wohlin [8] for this type of search. In addition between 2011 and 2012 Google Scholar has *"very significantly expanded its coverage... at a stable rate"* Harzing [26]. Primarily, we are concerned about getting a reliable number of citations for our analysis and not usability. Although we found it useful to reduce the number of papers to read manually due to the 'search within citing articles' feature. We are confident that Google Scholar is sufficient for our work.

Threats also exists in assessing and extracting information. We mitigated these threats; two authors in this study read and extracted information from 5 of the *final-set* of 39 papers and for the six factors extracted from all the 208 papers. Using the SLuRp tool Bowes et al. [20] any disagreements were identified and then resolved and the data updated.

The features of the data collected introduces another threat. In particular the analysis of citation count and the number of replications involves data with many ties. We therefore used Kendal's Tau correlation, rather than Spearman's correlation because it is known to deal with ties better.

We show that most threats have been minimised and believe to the best of our ability our findings are sound. We hope researchers replicate our study and our replication package is available (Online-Appendix). Under such conditions, significance tests of the Kendell correlation coefficient may be unreliable.

6. Discussion

Overall we have shown that defect prediction suffers from a lack of external replications with only 6% of 208 studies replicated. Silva et al. [4] identified 96 articles, reporting 133 replications performed between 1994 and 2010 in software engineering, indicating that replication in software engineering is carried out more frequently than in defect prediction. We also show that the few replications performed are not consistently systematic and of the 29 replications we analysed, only 18 (62%) results agreed with the original paper.

The characteristics of replicated original studies include those studies being published in the TSE journal and being based on closed source industrial data. Most of the replicated original studies do not satisfy a quality assessment ($quality_{4p}$) this despite being largely published in high impact venues. Such a potential lack of quality in original studies is surprising and suggests unreliable findings may be being propagated.

Reporting inconsistencies are also problematic for interpreting the outcomes of replications. For example, agreements are not always reported clearly, performance values of original studies are not always reported in the replication.

Our findings suggest that defect prediction replication can offer valuable lessons that can be built upon by others. The original studies that have multiple replications have demonstrated opportunities to improve defect prediction and develop more stable conclusions (e.g. (Org[5])).

Conversely, the lack of replication studies could be an indication of the need to define new research goals in defect prediction; our results may simply demonstrate decreasing interest in defect prediction. Recent criticisms of the area focus on the lack of impact that defect prediction research has in industry. For example, Lanza et al. [9] reported that the problem with defect prediction lies in how the approaches are evaluated and benchmarked and further suggested that *"researchers should seriously consider putting their predictors out into the real world and having them used by developers who work on a live code"*. Shepperd [27] mentioned that the evaluation of prediction models is problematic and *"that the concerns of researchers need to be better aligned with the likely end-users"*. Kitchenham [28] highlights the importance of these issues in relation to replication when she talked in-depth about the 4Rs (Rigour, Reproducibility, Replication and Relevance) and how they are linked; with good Rigour, there is value in Reproducing the work and also useful Replicating reproducible work to check stability across multiple organisations provided they are relevant to what the practitioners need. Kitchenham claims that *"very few papers consider practical issues"* [28] and suggests the need for obtaining more realistic datasets and collaborations with industry partners. It is criticisms related to these industry issues that are currently affecting the area of defect prediction.

Our results suggest that far more replications are needed. Furthermore that replications need to be done much more systematically. We show that important replication steps have been missed out based on the taxonomy that we applied. Incremental changes should be made to original studies in replications while analysing the effect of changes on model performance. Being systematic may be easier when artefacts are open source (for reuse and reduced variability) so that a researcher can break down a study into separate components. The typical components of a defect prediction study include, tools, statistics, cross validation, feature selection, parameter optimising, etc. (e.g. Table 5). Replicated experiments should be run with the same components as the original (reproduced), with intentional variation of changeable components implemented systematically, i.e. change one after another while recording their effect on model performance. This

systematic approach has the potential to discover those factors affecting results. A good example of systematic replication is Song et al. (Rep [16]); the study first reproduced the original Menzies et al. (Org[7]) to confirm it, then performed several combinations of the components while recording the effect on model performance.

6.1. Practical recommendations for the replication of defect prediction studies

We make the following recommendations for replication in defect prediction studies. These suggestions are not a hard and fast set of rules and as such should not be used as a mechanism to exclude papers from being replicated.

[Recommendation 1] Highly cited papers should be replicated as such papers tend to influence future defect prediction practice. Other papers should also be replicated.

[Recommendation 2] Use a replication infrastructure (e.g. OpenML [http://www.openml.org/] [29], or Zenodo [https://zenodo.org/]). Such infrastructures typically include an application programming interface API (Weka, R, REST, Java, .Net, Python, mlR, Moa) based repository designed to allow experiments to be configured on it and run on a user's machine. This keeps one version of datasets, the results, the protocol for easy sharing, and has persistence; most likely going to have the availability attribute [14] for researchers to use in future.

[Recommendation 3] Better use of existing reporting guidelines should be made. This requires the development of comprehensive software engineering reporting guidelines. These should be based on existing guidelines, including Runeson and Höst's [30] on case study design, Kitchenham et al.'s [31] on empirical software engineering, Carver's [32] on reporting replications, da Silva et al.'s [4] on designing and reporting replication studies and Mende's (Rep[9]) on replication remedies, pitfalls and challenges. Crucially, these guidelines must be collected and structured as a repository similar to the repositories that already exist in the Medical field (e.g. Munafò et al. [33, http://www.equator-network.org]).

[Recommendation 4] Replication Impact Factors should be put into practice. As Schimmack says: *"Demonstrating replicability should become an important criterion of research excellence that can be used by funding agencies and other stakeholders to allocate resources to research that advances science"* [34]. The following are possible ways in which replication can be implemented in impact factors:

- Use number of replications per study as additional impact factor metric $R-index$ [34].
- Use number of reproductions per study as additional impact factor metric $Repro_{index}$.
- Use number of replications and reproductions as the most significant impact factor metric RR_{index}.

[Recommendation 5] Quality assessments (e.g. $quality_{4p}$) should be applied to original studies. Researchers should consider quality in two parts; the quality of the methodology and quality of the reporting. These quality checks should be made on original studies before replication to minimise the spread of potentially erroneous results.

[Recommendation 6] The replication of important studies needs to be incentivised. Currently there is little reason for a researcher to replicate a study, as original studies are more likely to be cited than a replication. Highly rated publication venues should specifically encourage replications.

[Recommendation 7] Reproduction should be carried out before replication. This will demonstrate how close the replicating authors can get to the original study. There is little point attempting to replicate results if reproduction is not possible because, e.g. the raw defect data is not both accessible and held in a secure source.

These suggestions are not exhaustive. We hope that future researchers will evaluate, refine and extend these recommendations.

7. Conclusion

Replication is reported to be very important [6], yet not often enough performed in software engineering [4]. In this paper we particularly investigated replication in defect prediction - a very active area of research in software engineering. In this study we investigated the replication of 208 original defect prediction studies identified by a highly cited SLR [1].

Our findings suggest low replication in defect prediction and potential low quality in defect prediction studies. Only 13 of the 208 original studies have been replicated by researchers that are independent to those of the original studies. Only 3 of the 13 original replicated studies are assessed as quality studies with regards to research methodology and reporting. We have also shown some of the difficulty in comparing original results with replicated results, as replications can report their results using measures not used by original studies. This reporting inconsistency makes comparing results difficult.

We have given some practical suggestions to incentivise and standardise aspects of replication suggesting, for example the calculation of a new Replication and Reproduction impact factor, data sharing, and guidelines of reporting.

Our results show that studies published in a high impact journal (in particular TSE) tend to attract replications. This means that there is an opportunity that these publication venues could come up with ways to encourage more replications, for example a best replication paper award could be created. Industrial based original studies also seem to have more replications.

We hope our study drives discussions along the line of our suggestions and we hope researchers replicate and extend this study to get more insight into replication across Software Engineering.

Appendix A. Components that make up a defect prediction study

Table A.15
Changeable components of defect prediction studies adapted from [5].

Protocol	Operationalisation	Populations	Experimenter
Definition: The configuration of subcomponents to observe an outcome **Changes:** -Experimental design of how treatments are allocated, e.g. model building framework configures data preprocessing, parameter optimisation, cross validation, prediction, data collection framework configures defect linking, extracting and labelling -Statistical analyses	**Definition:** Mode of applying treatments (techniques) e.g. training a model on train and test set gives unrealistic results than on train set only **Changes (cause construct: cause of differences in results):** -Literature sources, training, instructions for applying a procedure during experiments -Tools used for running experiments e.g. IDE -Algorithms for, building prediction models, dealing with imbalance, linking defects **Changes (effect construct: effect on results):** -Defining dependent/ independent variables, e.g. number of defects post release/code complexity -Process of calculating the variables, e.g. number of defects fixed after released to customers and linked to the point the defect was introduced -Measuring model performance with different measures	**Definition:** The subject and objects properties used in a controlled experiment **Changes:** -Source code of project (open (Eclipse) or closed (NASA) source) -Design documents, programming language, size, complexity, maturity (years used and growth), domain etc. -Granularity of independent variables (metrics) such as class or method level, granularity of dependent variables such as defective or not and number of defects	**Definition:** The designer, trainer, monitor, measurer and analyst involved in the experiments (authors). **Changes:** -Different authors may or may not vary the parameters of an experiment on the same dataset

Appendix B. Changed components data extracted from replication studies

Table B.16
Protocol: Replication studies.

Rep. studies	Cross val.	Parameter Ttuning	Statistics	Data cleaning
Hongyu Zhang (Rep[3]), (Rep[4])	No	No	Computed the coefficient of determination and the Standard Error of Estimate	No
Ghotra et al. (Rep[8])	Yes	Yes	Scott–Knott statistical test	Mixed: Yes on NASA, No on Apache family
Leszak Marek (Rep[10])	No	No	No	No
Mende and Koschke (Rep[11])	10 by 10 cross validation	No	Friedman test, Nemenyi post hoc test	No
Galinac Grbac et al. (Rep[12])	NA	NA	Pearson correlation coefficient, nonparametric Spearman correlation, vote counting	Yes: removed duplicates, outliers
Devine et al. (Rep[13])	No	No	Spearman correlation	No
Hamill and Goseva-Popstojanova (Rep[2])				
Turhan and Bener (Rep[14])	10 by 10 randomised	No	*t*-test	No
Zhang et al. (Rep[15])	10 by 10 randomised	No	No	Removed duplicates, missing values
Song et al. Rep[16]	10 by 10 randomised	Yes	% difference, Wilcoxon signed 1-tailed	Removed outliers, missing values
Singh and Verma (Rep[17])	10 by 10 stratified randomised	No	No	No
Krishnan et al. (Rep[17])	10 by 1000 randomised	No	Figner–Killeen, Kruskal–Wallis, one-way ANOVA, *t*-test with Bonferroni correction of *p*-value	No
Rahman et al. (Rep[19])	No	No	Wilcoxon one sided paired	No
Tosun et al. (Rep[20])	split-sample 10 by 5	No	Spearman, Pearson	No
Nguyen et al. (Rep[21])	split-sample by 50	No	Spearman rank correlation, Wilcoxon rank test, ANOVA	No
Premraj and Herzig (Rep[22])	stratified hold-out	No	Kruskal–Wallis two pairs test, ANOVA	Yes
Okutan and Yıldız (Rep[23])	10 by 20 randomised stratified	No	*t*-test	Yes
Duala-Ekoko and Robillard (Rep[24])	No	No	Chi-square	No
Mende (Rep[9])	50 by 10fold and 10fold Cross-Val			
Kpodjedo et al. (Rep[25])	No	No	Wilcoxon signed rank test, Cohen-d statistics	No
Li et al. (Rep[26])	No	No	Weibull, Power, Gamma, Exponential, Theil	No

Table B.17
Operationalisation: Replication studies.

Rep. studies	Tools	Algorithms	Independent var	Dependent var
Hongyu Zhang (Rep[3]), (Rep[4])	SPSS	Non linearReg	Static code (complexity), structure of abstract syntax tree (no of nodes etc.)	Number of defects (prerelease, post-release).
Ghotra et al. (Rep[8])	Weka	Statistical, Clustering, Rule-based, Nearest Neighbours, NeuralNet, SVMs, Tree-based, Ensembles	Static code, CK, QMOOD, Martin's	Defective or Not defective
Leszak Marek (Rep[10])	ClearDDTSTM (from IBM-Rational), ClearCaseTM (from IBM-Rational)	Correlation analysis	Process, Complexity of changes, source file size, file age, defect density (file, release)	Number of defects (prerelease, post-release)
Mende and Koschke (Rep[11])	R	RandomForest	Static code metrics	Defective or Not defective
Galinac Grbac et al. (Rep[12])	No	No	Size (LOC)	Number of faults (pre-release, analyse post-release)
Devine et al. (Rep[13])	SourceMonitor (metrics), StatSVN (analyse SVN logs)	Stepwise regression	Source code, change, fault metrics	Number of defects, defect density
Hamill and Goseva-Popstojanova (Rep[1]) (Rep[2])	No	No	Fault types, detection activities, severity	Number of faults (prerelease, post-release)
Turhan and Bener (Rep[14])	Matlab: no version	Naive Bayes, Linear Discriminant, Quadratic Discriminant	Static code	Defective or Not defective
Zhang et al. (Rep[15])	Function to Component level data aggregator, Weka: No version.	BayesNet, Bagging, k-NN, RandFor, NeuralN, LogisticR, RBFNet, SVM, NaiveBayes, C4.5(J48), K-Star, AdaBoostM1	LOC, CyclComplex and HalsteadVol	Defective or Not defective
Song et al. Rep[16]	No	Naive Bayes, J48, OneR	Static code	Defective or Not defective
Singh and Verma (Rep[17])	No	K-means	Static code	Defective or Not defective
Krishnan et al. (Rep[18])	CVSPS(capture commit transactions), Weka, R	J48	Change	Defective, Not-defective
Rahman et al. (Rep[19])	Git	Least recently used (LRU)	Churn, temporal locality, spatial locality	defect density and the cost effectiveness of inspection
Tosun et al. (Rep[20])	Ucinet 6 Network Analysis tool (network metrics), open-source metrics extraction tool, Prest (dependencies)	NaiveBayes, LogisticReg, LinearReg	Complexity, network	Defective or Not defective
Nguyen et al. (Rep[21])	UCINET (network metrics), Structure101 (extract dependencies), Understand (complexity metrics)	linearReg, logisticReg	Import packages, Network metrics, Complexity, OO, Function	Number of faults(post-release), Defective or Not defective
Premraj and Herzig (Rep[22])	Understand V2.0 Build 505 (metrics for Java, C++), JDT frame (map class back to file), UCINET (network metrics), R	KNN, LogisticReg, NaiveBayes, Rpart, SVM, Tree-Bagging	Code, socio-technical (network metrics), combined	Number of defects, Defective or Not defective
Okutan and Yıldız (Rep[23])	PMD source code analyser plugin in Netbeans (for LOCQ), Weka (no version)	Bayesian Networks	OO, static code	Defective or Not defective, Number of defects
Duala-Ekoko and Robillard (Rep [24])	SemDiff and Mylyn (mapping bug fixes to bug reports)	No	Import packages, classes	Number of faults
Mende (Rep[9])	R	logisticReg, NaiveBayes	Process, change, entropy of change, entropy, churn of source code, source code metrics, CK, OO	Number of defects
Kpodjedo et al. (Rep[25])	PADL (extract CK metrics)	logistic regression	Design metrics (class diagrams)	Defective or Not defective
Li et al. (Rep[26])	R	PCA, linearReg, k-means, non-linearReg, CART	Change, development metrics (no of in-dev process defects)	Number of faults

Table B.18
Populations: Replication studies.

Rep. Studies	Prog. Lang.	Domain	Granularity	Source
Hongyu Zhang (Rep[3]), (Rep[4])	Java	IDE	Package	Eclipse Versions 2.0, 2.1, 3.0
Ghotra et al. (Rep[8])	C, Java	(Satellite, Flight, Storage), web	Module (predictions), code (metrics)	NASA, PROMISE, Apache, GNU
Leszak Marek (Rep[10])	C++, C, shell script, tcl/tk, perl	Telecom	File (metrics)	Lucent
Mende and Koschke (Rep[11])	R		Static code metrics	Defective or Not defective
Galinac Grbac et al. (Rep[12])	PLEX	Telecom	Module, file	Industrial 5 releases
Devine et al. (Rep[13])	Java	Software testing tools	Component	PolyFlow Software Product Line: 4 projects
Hamill and Goseva-Popstojanova (Rep[1]) (Rep[2])	No	Flight software safety-critical	Component	NASA (not public)
Turhan and Bener (Rep[14])	C, Java	Satellite, Flight, Storage	Method	NASA
Zhang (Rep[15])	C, Java	Satellite, Flight, Storage	Component	NASA
Song et al. Rep[16]	C, Java	Satellite, Flight, Storage	Method	NASA, PROMISE
Singh and Verma (Rep[17])	C, Java	Satellite, Flight, Storage	Method	NASA, PROMISE
Krishnan et al. (Rep[18])	Java, C, C++	IDE	File (predictions)	Eclipse 2.0, 2.1, 3.0, 3.3, 3.4, 3.5, 3.6
Rahman et al. (Rep[19])	C, Java	Web server, Image manipulator, File manager, Email client, Text search engine	File (predictions)	Apache Httpd, Gimp, Nautilus, Evolution, Lucene
Tosun et al. (Rep[20])	C, Java	IDE, Embedded systems	Function, file (predictions), file (metrics)	Eclipse, Embedded systems
Nguyen et al. (Rep[21])	Java	IDE	Class, package (predictions) Classes, Packages, Function (metrics)	Eclipse
Premraj and Herzig (Rep[22])	Java	IDE	Source files (predictions), class and method (metrics)	JRuby, ArgoUML, Eclipse
Okutan and Yıldız (Rep[23])	Java	Web Server	Class (predictions)	Apache Family
Duala-Ekoko and Robillard (Rep[24])	Java	IDE	File, Method (predictions)	Eclipse version 2,3
Mende (Rep[9])	Java	IDE	Class (predictions, metrics)	Eclipse (Mylyn, Equinox, PDE, Lucene, Score)
Kpodjedo et al. (Rep[25])	Java	IDE, UML, Javascript Interpreter	Class (predictions), Class diagram (metrics)	ArgoUML, Rhino
Li et al. (Rep[26])	No	Operating System	Release (predictions)	IBM

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.infsof.2018.02.003.

References

[1] T. Hall, S. Beecham, D. Bowes, D. Gray, S. Counsell, A systematic literature review on fault prediction performance in software engineering, Software Eng. IEEE Trans. 38 (6) (2012) 1276–1304, http://dx.doi.org/10.1109/TSE.2011.103.

[2] T. Hall, D. Bowes, The state of machine learning methodology in software fault prediction, Machine Learning and Applications (ICMLA), 2012 11th International Conference on, 2 (2012), pp. 308–313, http://dx.doi.org/10.1109/ICMLA.2012.226.

[3] D. Bowes, T. Hall, D. Gray, Dconfusion: a technique to allow cross study performance evaluation of fault prediction studies, Autom. Software Eng. 21 (2) (2014) 287–313, http://dx.doi.org/10.1007/s10515-013-0129-8.

[4] F.Q. Silva, M. Suassuna, A.C. França, A.M. Grubb, T.B. Gouveia, C.V. Monteiro, I.E. Santos, Replication of empirical studies in software engineering research: a systematic mapping study, Empir. Software Engg. 19 (3) (2014) 501–557, http://dx.doi.org/10.1007/s10664-012-9227-7.

[5] O.S. Gómez, N. Juristo, S. Vegas, Understanding replication of experiments in software engineering: a classification, Inf. Software Technol. 56 (8) (2014) 1033–1048.

[6] J.P. Ioannidis, Why most published research findings are false, PLoS Med. 2 (8) (2005) e124.

[7] R. Moonesinghe, M.J. Khoury, C.J. Janssens, Most published research findings are false–but a little replication goes a long way, PLoS Med. 4 (2) (2007).

[8] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, EASE '14, ACM, New York, NY, USA, 2014, pp. 38:1–38:10, http://dx.doi.org/10.1145/2601248.2601268.

[9] M. Lanza, A. Mocci, L. Ponzanelli, The tragedy of defect prediction, prince of empirical software engineering research, IEEE Software 33 (6) (2016) 102–105, http://dx.doi.org/10.1109/MS.2016.156.

[10] Z. Yu, N.A. Kraft, T. Menzies, How to read less: better machine assisted reading methods for systematic literature reviews, CoRR (2016). abs/1612.03224

[11] J.T. Leek, R.D. Peng, Opinion: reproducible research can still be wrong: adopting a prevention approach, Proc. Natl. Acad. Sci. 112 (6) (2015) 1645–1646, http://dx.doi.org/10.1073/pnas.1421412111.

[12] L. Madeyski, B. Kitchenham, Would wider adoption of reproducible research be beneficial for empirical software engineering research? J. Intell. Fuzzy Syst. 32 (2) (2017) 1509–1521, http://dx.doi.org/10.3233/JIFS-169146.

[13] M.J. Shepperd, D. Bowes, T. Hall, Researcher bias: the use of machine learning in software defect prediction, IEEE Trans. Software Eng. 40 (6) (2014) 603–616, http://dx.doi.org/10.1109/TSE.2014.2322358.

[14] J.M. González-Barahona, G. Robles, On the reproducibility of empirical software engineering studies based on data retrieved from development repositories, Empir. Software Eng. 17 (1) (2012) 75–89, http://dx.doi.org/10.1007/s10664-011-9181-9.

[15] G. Robles, Replicating msr: a study of the potential replicability of papers published in the mining software repositories proceedings, Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on, IEEE, 2010, pp. 171–180.

[16] S.N. Goodman, D. Fanelli, J.P.A. Ioannidis, What does research reproducibility mean? Sci. Transl. Med. 8 (341) (2016), http://dx.doi.org/10.1126/scitranslmed.aaf5027. 341ps12–341ps12

[17] D.W. Aksnes, Characteristics of highly cited papers, Res. Eval. 12 (3) (2003) 159–170, http://dx.doi.org/10.3152/147154403781776645.

[18] V. Garousi, J.M. Fernandes, Highly-cited papers in software engineering: the top-100, Inf. Software Technol. 71 (2016) 108–128, http://dx.doi.org/10.1016/j.infsof.2015.11.003.

[19] D. Fucci, G. Scanniello, S. Romano, M. Shepperd, B. Sigweni, F. Uyaguari, B. Turhan, N. Juristo, M. Oivo, An external replication on the effects of test-driven development using a multi-site blind analysis approach, Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '16, ACM, New York, NY, USA, 2016, pp. 3:1–3:10, http://dx.doi.org/10.1145/2961111.2962592.

[20] D. Bowes, T. Hall, S. Beecham, Slurp: A tool to help large complex systematic literature reviews deliver valid and rigorous results, Proceedings of the 2nd International Workshop on Evidential Assessment of Software Technologies, EAST '12, ACM, New York, NY, USA, 2012, pp. 33–36, http://dx.doi.org/10.1145/2372233.2372243.

[21] C. Marshall, P. Brereton, B. Kitchenham, Tools to support systematic reviews in software engineering: A feature analysis, Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, EASE '14, ACM, New York, NY, USA, 2014, pp. 13:1–13:10, http://dx.doi.org/10.1145/2601248.2601270.

[22] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The weka data mining software: an update, SIGKDD Explor. Newslett. 11 (1) (2009) 10–18, http://dx.doi.org/10.1145/1656274.1656278.

[23] H.F. Moed, Measuring contextual citation impact of scientific journals, J. Informetr. 4 (3) (2010) 265–277, http://dx.doi.org/10.1016/j.joi.2010.01.002.

[24] J. Petrić, D. Bowes, T. Hall, B. Christianson, N. Baddoo, The jinx on the nasa software defect data sets, Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering, EASE '16, ACM, New York, NY, USA, 2016, pp. 13:1–13:5, http://dx.doi.org/10.1145/2915970.2916007.

[25] M. Shepperd, Q. Song, Z. Sun, C. Mair, Data quality: some comments on the nasa software defect datasets, Software Eng. IEEE Trans. 39 (9) (2013) 1208–1215, http://dx.doi.org/10.1109/TSE.2013.11.

[26] A.-W. Harzing, A longitudinal study of google scholar coverage between 2012 and 2013, Scientometrics 98 (1) (2014) 565–575.

[27] M. Shepperd, Machine learning may be the answer but is it trustworthy?.

[28] B. Kitchenam, Back to basics- the 4r's of software estimation.

[29] J. Vanschoren, J.N. van Rijn, B. Bischl, L. Torgo, OpenML: networked science in machine learning, SIGKDD Explor. 15 (2) (2013) 49–60, http://dx.doi.org/10.1145/2641190.2641198.

[30] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empir. Software Eng. 14 (2) (2009) 131–164, http://dx.doi.org/10.1007/s10664-008-9102-8.

[31] B. Kitchenham, H. Al-Khilidar, M.A. Babar, M. Berry, K. Cox, J. Keung, F. Kurniawati, M. Staples, H. Zhang, L. Zhu, Evaluating guidelines for reporting empirical software engineering studies, Empir. Software Eng. 13 (1) (2008) 97–121.

[32] J.C. Carver, Towards reporting guidelines for experimental replications: a proposal, Citeseer, 2010.

[33] M.R. Munafò, B.A. Nosek, D.V.M. Bishop, K.S. Button, C.D. Chambers, N. Percie du Sert, U. Simonsohn, E.-J. Wagenmakers, J.J. Ware, J.P.A. Ioannidis, A manifesto for reproducible science, Nat. Hum. Behav. 1 (2017). 0021 EP – http://dx.doi.org/10.1038/s41562-016-0021

[34] U. Schimmack, Quantifying statistical research integrity: the replicability index, (2014), https://wordpress.com/post/replication-index.wordpress.com/920.

Original Studies

[Org1] M. D'Ambros, M. Lanza, R. Robbes, An extensive comparison of bug prediction approaches, Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on, (2010), pp. 31–41, http://dx.doi.org/10.1109/MSR.2010.5463279.

[Org2] T. Zimmermann, N. Nagappan, Predicting defects using network analysis on dependency graphs, Proceedings of the 30th International Conference on Software Engineering, ICSE '08, ACM, New York, NY, USA, 2008, pp. 531–540, http://dx.doi.org/10.1145/1368088.1368161.

[Org3] T. Zimmermann, R. Premraj, A. Zeller, Predicting defects for eclipse, Predictor Models in Software Engineering, 2007. PROMISE'07: ICSE Workshops 2007. International Workshop on, IEEE, 2007, p. 9.

[Org4] C. Andersson, P. Runeson, A replicated quantitative analysis of fault distributions in complex software systems, software engineering, IEEE Trans. 33 (5) (2007) 273–286, http://dx.doi.org/10.1109/TSE.2007.1005.

[Org5] S. Lessmann, B. Baesens, C. Mues, S. Pietsch, Benchmarking classification models for software defect prediction: a proposed framework and novel findings, software engineering, IEEE Trans. 34 (4) (2008) 485–496, http://dx.doi.org/10.1109/TSE.2008.35.

[Org6] T. Ostrand, E. Weyuker, R. Bell, Predicting the location and number of faults in large software systems, software engineering, IEEE Trans. 31 (4) (2005) 340–355, http://dx.doi.org/10.1109/TSE.2005.49.

[Org7] T. Menzies, J. Greenwald, A. Frank, Data mining static code attributes to learn defect predictors, Software Eng. IEEE Trans. 33 (1) (2007) 2–13, http://dx.doi.org/10.1109/TSE.2007.256941.

[Org8] N. Fenton, N. Ohlsson, Quantitative analysis of faults and failures in a complex software system, Software Eng. IEEE Trans. 26 (8) (2000) 797–814, http://dx.doi.org/10.1109/32.879815.

[Org9] R. Moser, W. Pedrycz, G. Succi, A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction, Software Engineering, 2008. ICSE '08. ACM/IEEE 30th International Conference on, (2008), pp. 181–190, http://dx.doi.org/10.1145/1368088.1368114.

[Org10] S. Kim, T. Zimmermann, E.J. Whitehead Jr., A. Zeller, Predicting faults from cached history, Proceedings of the 29th International Conference on Software Engineering, ICSE '07, IEEE Computer Society, Washington, DC, USA, (2007), pp. 489–498, http://dx.doi.org/10.1109/ICSE.2007.66.

[Org11] S. Amasaki, Y. Takagi, O. Mizuno, T. Kikuno, A bayesian belief network for assessing the likelihood of fault content, Software Reliability Engineering, 2003. ISSRE 2003. 14th International Symposium on, (2003), pp. 215–226, http://dx.doi.org/10.1109/ISSRE.2003.1251044.

[Org12] A. Schröter, T. Zimmermann, A. Zeller, Predicting component failures at design time, Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering, ACM, 2006, pp. 18–27.

[Org13] T. Khoshgoftaar, N. Seliya, Tree-based software quality estimation models for fault prediction, Software Metrics, 2002. Proceedings. Eighth IEEE Symposium on, (2002), pp. 203–214, http://dx.doi.org/10.1109/METRIC.2002.1011339.

Replicated Studies

[Rep1] M. Hamill, K. Goseva-Popstojanova, Exploring fault types, detection activities, and failure severity in an evolving safety-critical software system, Software Qual. J. 23 (2) (2015) 229–265, http://dx.doi.org/10.1007/s11219-014-9235-5.

[Rep2] M. Hamill, K. Goseva-Popstojanova, Exploring fault types, detection activities, and failure severity in an evolving safety-critical software system, Software Qual. J. 23 (2) (2015) 229–265, http://dx.doi.org/10.1007/s11219-014-9235-5.

[Rep3] H. Zhang, On the distribution of software faults, Software Eng. IEEE Trans. 34 (2) (2008) 301–302, http://dx.doi.org/10.1109/TSE.2007.70771.

[Rep4] H. Zhang, On the distribution of software faults, Software Eng. IEEE Trans. 34 (2) (2008) 301–302, http://dx.doi.org/10.1109/TSE.2007.70771.

[Rep5] C. Andersson, P. Runeson, A replicated quantitative analysis of fault distributions in complex software systems, Software Eng. IEEE Trans. 33 (5) (2007) 273–286, http://dx.doi.org/10.1109/TSE.2007.1005.

[Rep6] S. Lessmann, B. Baesens, C. Mues, S. Pietsch, Benchmarking classification models for software defect prediction: a proposed framework and novel findings, Software Eng. IEEE Trans. 34 (4) (2008) 485–496, http://dx.doi.org/10.1109/TSE.2008.35.

[Rep7] T. Ostrand, E. Weyuker, R. Bell, Predicting the location and number of faults in large software systems, Software Eng. IEEE Trans. 31 (4) (2005) 340–355, http://dx.doi.org/10.1109/TSE.2005.49.

[Rep8] B. Ghotra, S. McIntosh, A.E. Hassan, Revisiting the impact of classification techniques on the performance of defect prediction models, Proc. of the 37th Int'l Conf. on Software Engineering (ICSE), (2015).

[Rep9] T. Mende, Replication of defect prediction studies: Problems, pitfalls and recommendations, Proceedings of the 6th International Conference on Predictive Models in Software Engineering, PROMISE '10, ACM, New York, NY, USA, 2010, pp. 5:1–5:10, http://dx.doi.org/10.1145/1868328.1868336.

[Rep10] M. Leszak, Software defect analysis of a multi-release telecommunications system, Product Focused Software Process Improvement, Springer, 2005, pp. 98–114.

[Rep11] T. Mende, R. Koschke, Effort-aware defect prediction models, Software Maintenance and Reengineering (CSMR), 2010 14th European Conference on, (2010), pp. 107–116, http://dx.doi.org/10.1109/CSMR.2010.18.

[Rep12] T.G. Grbac, P. Runeson, D. Huljenić, A second replicated quantitative analysis of fault distributions in complex software systems, Software Eng. IEEE Trans. 39 (4) (2013) 462–476, http://dx.doi.org/10.1109/TSE.2012.46.

[Rep13] T.R. Devine, K. Goseva-Popstojanova, S. Krishnan, R.R. Lutz, J.J. Li, An empirical study of pre-release software faults in an industrial product line, 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation, (2012), pp. 181–190, http://dx.doi.org/10.1109/ICST.2012.98.

[Rep14] B. Turhan, A. Bener, A multivariate analysis of static code attributes for defect prediction, Quality Software, 2007. QSIC '07. Seventh International Conference on, (2007), pp. 231–237, http://dx.doi.org/10.1109/QSIC.2007.4385500.

[Rep15] H. Zhang, X. Zhang, M. Gu, Predicting defective software components from code complexity measures, Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium on, (2007), pp. 93–96, http://dx.doi.org/10.1109/PRDC.2007.28.

[Rep16] Q. Song, Z. Jia, M. Shepperd, S. Ying, J. Liu, A general software defect-proneness prediction framework, Software Eng. IEEE Trans. 37 (3) (2011) 356–370, http://dx.doi.org/10.1109/TSE.2010.90.

[Rep17] P. Singh, S. Verma, An efficient software fault prediction model using cluster based classification, Int. J. Appl. Inf. Syst. (IJAIS) 7 (3) (2014) 35–41.

[Rep18] S. Krishnan, C. Strasburg, R.R. Lutz, K. Goseva-Popstojanova, K.S. Dorman, Predicting failure-proneness in an evolving software product line, Inf. Software Technol. 55 (8) (2013) 1479–1495, http://dx.doi.org/10.1016/j.infsof.2012.11.008. http://www.sciencedirect.com/science/article/pii/S0950584912002340 .

[Rep19] F. Rahman, D. Posnett, A. Hindle, E. Barr, P. Devanbu, Bugcache for inspections: hit or miss? Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, ESEC/FSE '11, ACM, New York, NY, USA, 2011, pp. 322–331, http://dx.doi.org/10.1145/2025113.2025157.

[Rep20] A. Tosun, B. Turhan, A. Bener, Validation of network measures as indicators of defective modules in software systems, Proceedings of the 5th International Conference on Predictor Models in Software Engineering, PROMISE '09, ACM, New York, NY, USA, 2009, pp. 5:1–5:9, http://dx.doi.org/10.1145/1540438.1540446.

[Rep21] T. Nguyen, B. Adams, A. Hassan, Studying the impact of dependency network measures on software quality, Software Maintenance (ICSM), 2010 IEEE International Conference on, (2010), pp. 1–10, http://dx.doi.org/10.1109/ICSM.2010.5609560.

[Rep22] R. Premraj, K. Herzig, Network versus code metrics to predict defects: areplication study, Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on, (2011), pp. 215–224, http://dx.doi.org/10.1109/ESEM.2011.30.

[Rep23] A. Okutan, O.T. Yıldız, Software defect prediction using bayesian networks, Empir. Software Eng. 19 (1) (2014) 154–181.

[Rep24] E. Duala-Ekoko, M.P. Robillard, A detailed examination of the correlation between imports and failure-proneness of software components, Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM '09, IEEE Computer Society, Washington, DC, USA, (2009), pp. 34–43, http://dx.doi.org/10.1109/ESEM.2009.5316047.

[Rep25] S. Kpodjedo, F. Ricca, P. Galinier, Y.G. Guéhéneuc, G. Antoniol, Design evolution metrics for defect prediction in object oriented systems, Empir. Software Eng. 16 (1) (2011) 141–175, http://dx.doi.org/10.1007/s10664-010-9151-7.

[Rep26] P.L. Li, M. Shaw, J. Herbsleb, P. Santhanam, B. Ray, An Empirical Comparison of Field Defect Modeling Methods, (2005).

A.2 Publication: What is the Impact of Imbalance on Software Defect Prediction Performance?

What is the Impact of Imbalance on Software Defect Prediction Performance?

Zaheed Mahmood
School of Technology
Research Institute
Hatfield, Hertfordshire.
AL10 9AB, United Kingdom.
z.mahmood4@herts.ac.uk

David Bowes
School of Technology
Research Institute
Hatfield, Hertfordshire.
AL10 9AB, United Kingdom.
d.h.bowes@herts.ac.uk

Peter C. R. Lane
School of Technology
Research Institute
Hatfield, Hertfordshire.
AL10 9AB, United Kingdom.
peter.lane@bcs.org.uk

Tracy Hall
Dept. of Computer Science
Brunel University
Uxbridge, Middlesex, London.
UB8 3PH, United Kingdom.
tracy.hall@brunel.ac.uk

ABSTRACT

Software defect prediction performance varies over a large range. Menzies suggested there is a ceiling effect of 80% *Recall* [8]. Most of the data sets used are highly imbalanced. This paper asks, what is the empirical effect of using different datasets with varying levels of imbalance on predictive performance? We use data synthesised by a previous meta-analysis of 600 fault prediction models and their results. Four model evaluation measures (the Mathews Correlation Coefficient (*MCC*), *F-Measure*, *Precision* and *Recall*) are compared to the corresponding data imbalance ratio. When the data are imbalanced, the predictive performance of software defect prediction studies is low. As the data become more balanced, the predictive performance of prediction models increases, from an average *MCC* of 0.15, until the minority class makes up 20% of the instances in the dataset, where the *MCC* reaches an average value of about 0.34. As the proportion of the minority class increases above 20%, the predictive performance does not significantly increase. Using datasets with more than 20% of the instances being defective has not had a significant impact on the predictive performance when using *MCC*. We conclude that comparing the results of defect prediction studies should take into account the imbalance of the data.

Keywords

Defect Prediction, Machine Learning, Data Imbalance

1. INTRODUCTION

PROMISE '15, October 21 2015, Beijing, China

DOI: http://dx.doi.org/10.1145/2810146.2810150

Defects in software cost the software engineering industry in excess of $50billion per year to put right in the US [7, 10]. In order to reduce the number of defects which are released, software engineers use different techniques to identify where defects are. Techniques for identifying defects include manual inspection and unit testing. Over the last fifteen years regression and machine learning techniques have been used to predict where defects may be in the code. Software defect prediction should allow development teams to allocate resources more effectively to testing and code reviews, thereby improving software quality and reliability [5].

A recent meta-analyses of 42 primary studies [11] identified 4 major factors that influence model performance (see Table 1).

Factor	Partial η^2
Researcher Group	31.01%
Dataset Family	31.00%
Input Metrics	12.44%
Classifier Family	8.23%

Table 1: The different factors which may affect the predictive performance of a defect prediction study and the proportion of the variance which the factor would account for if a prediction model is built using just one factor at a time (Partial η^2) [11]

Shepperd *et al.* [11] found that over 40 percent of data sets used by the 42 studies have less than 20 percent instances of the positive class (i.e., defect-prone). Highly imbalanced datasets are well-known to reduce the ability of a machine learning algorithm to predict the infrequent class. The explanation lies in the preference of many algorithms for the 'simplest' hypothesis, as apparent in decision-tree algorithms and support-vector machines; this simplest hypothesis is liable to ignore a minority class with relatively few representatives. Several approaches have been developed to handle imbalanced data. The more popular modify the training set and include: over-sampling, under-sampling and synthetic minority over-sampling technique (SMOTE [3]).

When applied to imbalanced datasets, these techniques have been shown, in general, to increase the predictive performance of standard classifiers [1, 6, 9, 12]. However, rebalancing can be misapplied. Blagus [2] shows that SMOTE does not significantly improve the performance of classifiers when more than 40 independent variables are used. Lane *et al.* [6] found that balancing the data in a sentiment analysis task had the most impact on severe imbalances, but was not so effective when the minority class made up 25% or more of the data; also, balancing the data for algorithms which can adapt to different class probabilities, such as Naïve Bayes, can be less effective.

In this paper, we conduct an exploratory study of the impact imbalance has on the performance of defect prediction models. We hypothesise that the performance of models increases as the proportion of the minority class approaches 50% of the population.

The remainder of this paper is structured as: Section 2 describes the methodology. Section 3 presents our preliminary findings. Section 4 concludes with our recommendations.

2. METHODOLOGY

To test the assumption that moving to a more balanced dataset will increase predictive performance, we first determine if imbalance has a significant impact on predictive performance. We extracted the frequency of the defective class (*balance*) of the datasets used for the 600 models in [11].

The data from [11] are based on defect prediction studies which have predicted a module of code as being either defective or not defective. When a model is trained to predict an instance as being defective (POSITIVE) or not defective (NEGATIVE) the results can be summarised in a confusion matrix (see Table 2). Compound performance measures can be computed from the confusion matrix to reveal different properties of the prediction models (see Table 3). We extract the following binary classification measures, *MCC*, *F-Measure*, *Precision* and *Recall*, from the extracted confusion matrices. We chose *MCC* because it combines all four quadrants of the confusion matrix and does not ignore the many true negatives; *Precision* because high values indicate that few predictions are wrong and a developer would not be wasting their time investigating these predictions; *Recall* shows the proportion of defects actually predicted, which is important in safety critical systems; and *F-Measure* combines both *Precision* and *Recall*.

	Observed Positive	Observed Negative
Predicted Positive	True Positives (TP)	False Positives (FP)
Predicted Negative	False Negatives (FN)	True Negatives (TN)

Table 2: A Binary Confusion Matrix

We carried out a univariate analysis of variance by converting the numerical *balance* into 20 levels ($balance_5$), e.g., level 1 is any imbalance in the range 0.000 to 0.049. The analysis of variance used a random effects model to show if imbalance was a significant factor rather than which level(s) are significant. Table 4 shows that $balance_5$ does contribute significantly to the variance in the results, however, dataset family (from which imbalance is derived) contributes more. In conclusion, imbalance of the dataset is important, but dataset family has other features (beyond the scope of this study) which are causing more variations in the performance of defect prediction studies.

	Partial η^2	Pr(>F)
Researcher Group	31.01%	< 0.0001
Dataset Family	31.00%	< 0.0001
$balance_5$	**18.76%**	< 0.0001
Input Metrics	12.44%	< 0.0001
Classifier Family	8.23%	< 0.0001

Table 4: Table showing how the variance result from $balance_5$ compares with other possible factors.

Having established that data imbalance can be a significant factor in defect prediction, we now investigate the nature of the relationship between Predictive performance and imbalance. We create scatterplots of *F-Measure*, *Precision* and *Recall* against *balance*.

3. RESULTS

Figure 1 shows that, as *balance* initially increases, *MCC* also increases. This means that datasets with very few defects tend to result in prediction models which are poor at predicting defects. There is a change in *MCC* from 0.15 to 0.35 when *balance* changes from 0.00 to 0.20. Figure 1 has few data points where $0.21 < balance < 0.30$ and, although average predictive performance appears to increase, we can not be confident in saying that performance increases up to $balance = 0.3$. The general increase is also observed for *F-Measure* and *Precision*. It is interesting to note that *Recall* initially decreases as *balance* increases.

As *balance* increases from 0.3 to 0.5 (maximum level of balance), the predictive performance does not tend to increase for *MCC*. Predictive performance as measured by *Precision*, *Recall* and *F-Measure* does increase.

4. CONCLUSION

Data imbalance is a factor which affects software defect prediction. To some extent this is not a surprise as imbalance has been observed to affect the ability to make predictions in other fields [6]. What is important is that the evidence from published work shows that performance improves as *balance* increases from 0.0 to 0.2 with the exception of *Recall*. As *balance* changes from 0.2 to 0.5. the story is different, *MCC* no longer appears to improve. Other performance measures do increase but not as rapidly.

Why does this matter? Because when we clean or modify our data we implicitly change the imbalance of the dataset [4]. This preliminary study suggests that we can estimate the amount by which *MCC* may increase for datasets with a low *balance*. Correcting the predictive performance for changes in imbalance as a result of cleaning allows us to see if an improvement in predictive performance is due to the cleaning step or the change in imbalance. It is also interesting to note that, as *balance* initially increases, *Recall* declines. If a cleaning step does not cause *Recall* to decline, we may be able to conclude that the cleaning step may not be the cause of an improvement in predictive performance.

5. ACKNOWLEDGEMENTS

Figure 1: Scatterplot of predictive performance against dataset *balance*. Showing that for *MCC*, as the *balance* increase from 0.00 to 0.20, the *MCC* increases. An increase of *balance* from 0.20 to 0.5 does not seem to change *MCC*. For the first scatterplot, we identify a small number of possible outliers. We have also plotted the convex-hull of 'best' performance as *balance* increases.

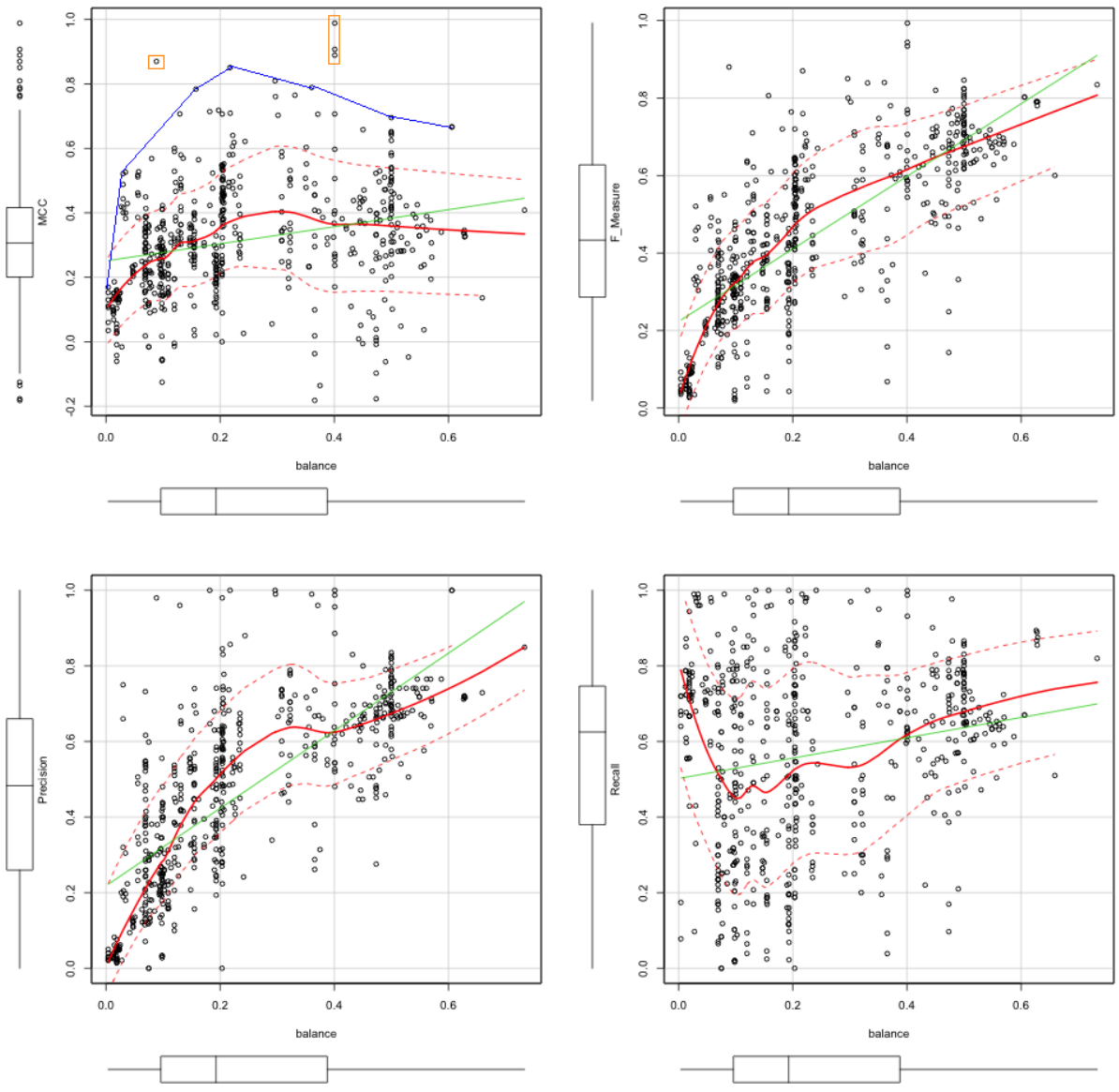


Table 3: Compound Performance Measures from a Binary Confusion Matrix

Measures	Defined As	Meaning
Recall	$\frac{TP}{TP+FN}$	Proportion of actual positives found.
Precision	$\frac{TP}{TP+FP}$	Proportion of predicted positives which are true positives
F-Measure	$\frac{2 \times Recall \times Precision}{Recall + Precision} = \frac{2TP}{2TP+FP+FN}$	The harmonic mean of *Precision* and *Recall*.
Matthews Correlation Coefficient (*MCC*)	$\frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$	A regression coefficient using all four quadrants of a confusion matrix.

This work was partly funded by a grant from the UK's Engineering and Physical Sciences Research Council under grant number: EP/L011751/1

References

[1] G. Batista, D. Silva, and R. Prati. An experimental design to evaluate class imbalance treatment methods. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 95–101. IEEE, 2012.

[2] R. Blagus and L. Lusa. Evaluation of smote for high-dimensional class-imbalanced microarray data. In *Machine learning and applications (icmla), 2012 11th international conference on*, volume 2, pages 89–94. IEEE, 2012.

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, pages 321–357, 2002.

[4] D. P. H. Gray. *Software defect prediction using static code metrics:formulating a methodology*. PhD thesis, University of Hertfordshire, 2013.

[5] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell. A systematic literature review on fault prediction performance in software engineering. *Software Engineering, IEEE Transactions on*, 38(6):1276–1304, Nov 2012.

[6] P. C. R. Lane, D. Clarke, and P. Hender. On developing robust models for favourability analysis: Model choice, feature sets and imbalanced data. *Decision Support Systems*, 53(4):712–718, 2012.

[7] M. Levinson. Let's stop wasting $78 billion a year'. *CIO, 15th October*, pages 78–83, 2001.

[8] T. Menzies, B. Turhan, A. Bener, G. Gay, B. Cukic, and Y. Jiang. Implications of ceiling effects in defect predictors. In *Proceedings of the 4th international workshop on Predictor models in software engineering*, pages 47–54. ACM, 2008.

[9] D. Rodriguez, I. Herraiz, R. Harrison, J. Dolado, and J. C. Riquelme. Preliminary comparison of techniques for dealing with imbalance in software defect prediction. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14, pages 43:1–43:10, New York, NY, USA, 2014. ACM.

[10] P. Runeson and A. Andrews. Detection or isolation of defects? an experimental comparison of unit testing and code inspection. In *Software Reliability Engineering, 2003. ISSRE 2003. 14th International Symposium on*, pages 3–13. IEEE, 2003.

[11] M. Shepperd, D. Bowes, and T. Hall. Researcher bias: The use of machine learning in software defect prediction. *Software Engineering, IEEE Transactions on*, 40(6):603–616, June 2014.

[12] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano. Experimental perspectives on learning from imbalanced data. In V. Ghahramani, editor, *Proceedings of the 24th international conference on Machine learning*, pages 935–942. ACM, 2007.

Table A.1: Protocol: Replication Studies

Rep. Studies	Cross Val.	Parameter Tuning	Statistics	Data Cleaning
Zhang [2008a],Zhang [2008b]	No	No	computed the coefficient of determination and the Standard Error of Estimate	No
Ghotra et al. [2015]	Yes	Yes	Scott-Knott statistical test	Mixed: Yes on NASA, No on Apache family
Leszak [2005]	No	No	No	No
Mende and Koschke [2010]	10 by 10 cross validation	No	Friedman test, Nemenyi post hoc test	No
Galinac Grbac et al. [2013]	NA	NA	Pearson correlation coefficient, nonparametric Spearman correlation, vote counting	Yes: removed duplicates, outliers
Devine et al. [2012]	No	No	Spearman correlation	No
Hamill and Goseva-Popstojanova [2015b]				
Turhan and Bener [2007]	10 by 10 randomised	No	T-test	No
Zhang et al. [2007]	10 by 10 randomised	No	No	Removed duplicates, missing values
Song et al. [2011]	10 by 10 randomised	Yes	% difference, Wilcoxin signed 1-tailed	Removed outliers, missing values
Singh and Verma [2014]	10 by 10 stratified randomised	No	No	No
Krishnan et al. [2013]	10 by 1000 randomised	No	Figner-Killeen, Kruskal-Wallis, one-way ANOVA, t-test with Bonferroni correction of p-value	No
Rahman et al. [2011]	No	No	Wilcoxon one sided paired	No
Tosun et al.	split-sample 10 by 5	No	Spearman, Pearson	No
Nguyen et al. [2010]	split-sample by 50	No	Spearman rank correlation, Wilcoxon rank test, ANOVA	No
Premraj and Herzig [2011]	stratified hold-out	No	Kruskal-Wallis two pairs test, ANOVA	Yes
Okutan and Yıldız [2014]	10 by 20 randomised stratified	No	t-test	Yes
Duala-Ekoko and Robillard [2009]	No	No	Chi-square	No
Mende [2010]	50 by 10fold and 10fold Cross-Val			
Kpodjedo et al. [2011]	No	No	Wilcoxon signed rank test, Cohen-d statistics	No
Li et al. [2005]	No	No	Weibull, Power, Gamma, Exponential, Theil	No

A.3 Changed components data extracted from Replication Studies

Table A.2: Operationalisation: Replication Studies

Rep. Studies	Tools	Algorithms	Independent Var	Dependent Var
Zhang [2008a], RepZhang [2008b]	SPSS	Non linearReg	static code (complexity), structure of abstract syntax tree (no of nodes etc.)	Number of defects (prerelease, postrelease).
Ghotra et al. [2015]	Weka	Statistical, Clustering, Rule-based, Nearest Neighbours, NeuralNet, SVMs, Tree-based, Ensembles	static code, CK, QMOOD, Martin's	Defective or Not defective
Leszak [2005]	ClearDDTSTM (from IBM-Rational), ClearCaseTM (from IBM-Rational)	Correlation analysis	Process, Complexity of changes, source file size, file age, defect density (file, release)	Number of defects (prerelease, postrelease)
Mende and Koschke [2010]	R	RandomForest	Static code metrics	Defective or Not defective
Galinac Grbac et al. [2013]	No	No	Size (LOC)	Number of faults (pre-release, analyse post-release)
Devine et al. [2012]	SourceMonitor (metrics), StatSVN (analyse SVN logs)	Stepwise regression	source code, change, fault metrics	Number of defects, defect density
Hamill and Goseva-Popstojanova [2015a] RepHamill and Goseva-Popstojanova [2015b]	No	No	fault types, detection activities, severity	Number of faults (prerelease, postrelease)
Turhan and Bener [2007]	Matlab: no version	Naive Bayes, Linear Discriminant, Quadratic Discriminant	static code	Defective or Not defective
Zhang et al. [2007]	Function to Component level data aggregator, Weka: No version.	BayesNet, Bagging, k-NN, RandFor, NeuralN, LogisticR, RBFNet, SVM, NaiveBayes, C4.5(J48), K-Star, AdaBoostM1	LOC, CyclComplex and HalsteadVol	Defective or Not defective
Song et al. [2011]	No	Naive Bayes, J48, OneR	static code	Defective or Not defective
Singh and Verma [2014]	No	K-means	static code	Defective or Not defective
Krishnan et al. [2013]	CVSPS(capture commit transactions), Weka, R	J48	change	Defective, Not-defective
Rahman et al. [2011]	Git	Least recently used (LRU)	churn, temporal locality, spatial locality	defect density and the cost effectiveness of inspection
Tosun et al.	Ucinet 6 Network Analysis tool (network metrics), open-source metrics extraction tool, Prest (dependencies)	NaiveBayes, LogisticReg, LineaReg	complexity, network	Defective or Not defective
Nguyen et al. [2010]	UCINET (network metrics), Structure101 (extract dependencies), Understand (complexity metrics)	linearReg, logisticReg	Import packages, Network metrics, Complexity, OO, Function	Number of faults(postrelease), Defective or Not defective
Premraj and Herzig [2011]	Understand V2.0 Build 505 (metrics for Java, C++), JDT frame (map class back to file), UCINET (network metrics), R	KNN, LogisticReg, NaiveBayes, Rpart, SVM, TreeBagging	code, socio-technical (network metrics), combined	Number of defects, Defective or Not defective
Okutan and Yıldız [2014]	PMD source code analyser plugin in Netbeans (for LOCQ), Weka (no version)	Bayesian Networks	OO, static code	Defective or Not defective, Number of defects
Duala-Ekoko and Robillard [2009]	SemDiff and Mylyn (mapping bug fixes to bug reports)	No	Import packages, classes	Number of faults
Mende [2010]	R	logisticReg, NaiveBayes	process, change , entropy of change , entropy, churn of source code, source code metrics, CK, OO	Number of defects
Kpodjedo et al. [2011]	PADL (extract CK metrics)	logistic regression	Design metrics (class diagrams)	Defective or Not defective
Li et al. [2005]	R	PCA, linearReg, k-means, nonlinearReg, CART	change, development metrics (no of in-dev process defects)	Number of faults

Table A.3: Populations: Replication Studies

Rep. Studies	Prog. Lang.	Domain	Granularity	Source
Zhang [2008a], RepZhang [2008b]	Java	IDE	package	Eclipse Versions 2.0, 2.1, 3.0
Ghotra et al. [2015]	C, Java	(Satellite, Flight, Storage), web	Module (predictions), code (metrics)	NASA, PROMISE, Apache, GNU
Leszak [2005]	C++, C, shell script, tcl/tk, perl	Telecom	File (metrics)	Lucent
Mende and Koschke [2010]	R		Static code metrics	Defective or Not defective
Galinac Grbac et al. [2013]	PLEX	Telecom	Module, file	Industrial 5 releases
Devine et al. [2012]	Java	Software testing tools	Component	PolyFlow Software Product Line: 4 projects
Hamill and Goseva-Popstojanova [2015a] Hamill and Goseva-Popstojanova [2015b]	No	Flight software safety-critical	Component	NASA (not public)
Turhan and Bener [2007]	C , Java	Satellite, Flight, Storage	Method	NASA
Zhang et al. [2007]	C, Java	Satellite, Flight, Storage	Component	NASA
Song et al. [2011]	C, Java	Satellite, Flight, Storage	Method	NASA, PROMISE
Singh and Verma [2014]	C, Java	Satellite, Flight, Storage	Method	NASA, PROMISE
Krishnan et al. [2013]	Java, C, C++	IDE	File (predictions)	Eclipse 2.0, 2.1, 3.0, 3.3, 3.4, 3.5, 3.6
Rahman et al. [2011]	C, Java	Web server, Image manipulator, File manager, Email client, Text search engine	File (predictions)	Apache Httpd, Gimp, Nautilus, Evolution, Lucene
Tosun et al.	C, Java	IDE, Embedded systems	function, file (predictions), file (metrics)	Eclipse, Embedded systems
Nguyen et al. [2010]	Java	IDE	Class, package (predictions) Classes, Packages, Function (metrics)	Eclipse
Premraj and Herzig [2011]	Java	IDE	source files (predictions), class and method (metrics)	JRuby, ArgoUML, Eclipse
Okutan and Yıldız [2014]	Java	Web Server	Class (predictions)	Apache Family
Duala-Ekoko and Robillard [2009]	Java	IDE	File, Method (predictions)	Eclipse version 2,3
Mende [2010]	Java	IDE	Class (predictions,metrics)	Eclipse (Mylyn, Equinox, PDE, Lucene, Score)
Kpodjedo et al. [2011]	Java	IDE, UML, Javascript Interpreter	Class (predictions), Class diagram (metrics)	ArgoUML, Rhino
Li et al. [2005]	No	Operating System	Release (predictions)	IBM

A.4 Synthesised agreements determined in replications studies

Table A.4: 13 replicated original studies out of the 208 with their replication studies, replication agreements between studies (this table is not the same as Table 4.1 and 4.11). This Table contains the synthesised agreements between originals and their replications showing where the agreements were determined. All values and text of the agreements are copied as written in the replication papers.

Replicated original studies	Replication studies	Rep. type & agreement answers	Corresponding agreement data synthesised from the replication papers
[D'Ambros et al. 2010]	[Mende 2010]	(A) Yes	"our results and the original ones by D'ambros et al.,... is remarkable low: The mean difference for adjusted R2 between our results is 0.00621, and 0.0030 for Spearman's ρ".
		(G) Yes	"the differences were negligible. The mean difference in our case is 0.0023 for adjusted R2 and 0.0028 for ρ"
[Andersson and Runeson 2007]	[Hamill and Goseva-Popstojanova 2015a]	(H) Yes	"Results are consistent with some works (..., Andersson and Runeson, 2007")
	[Zhang 2008a]	(H) No	"We have replicated the studies in [1] and [3] and discovered that the distribution of faults over modules can be better modeled using the Weibull probability distribution function". [3] is the original study here.
[Lessmann et al. 2008]	[Ghotra et al. 2015]	(A) Yes	"Our replication yields...similar results to the prior study, i.e., the impact that classification techniques have appear to be minimal."
		(H) No	"we found that the results differ from those of Lessmann et al. [34], i.e., classification techniques produce defect prediction models with significantly different performance."
[Ostrand et al. 2005]	[Leszak 2005]	Partial (H)	"In our trial to replicate their study, we succeeded to extract all the raw data their studies are based upon. Looking on the evaluation results, we could confirm some of their results; other results differ significantly"
	[Mende and Koschke 2010]	Unknown (G)	—
[Fenton and Ohlsson 2000]	[Andersson and Runeson 2007]	Partial (H)	"a small number of modules contain a majority of the faults (confirmed), 2) fault persistence between test phases (high fault incidence in function testing), 3) the relation between number of faults and lines of code (neither confirmed nor disproved), and 4) fault density similarities across test phases and projects (confirmed)".
	[Galinac Grbac et al. 2013]	Partial (H)	"Pareto principle is clearly confirmed,..., Size-related predictors, on the other hand, are not given any support,..., fault density across releases and environments is of the same magnitude, but still varies a lot"
	[Ostrand et al. 2005]	Yes (H)	"the 20 percent of the files with the highest predicted number of faults contained between 71 percent and 92 percent of the faults that were actually detected"
	[Zhang 2008b]	No (H)	"We have replicated the studies in [1] and [3] and discovered that the distribution of faults over modules can be better modeled using the Weibull probability distribution function". [1] is the original study here.
	[Devine et al. 2012]	Yes (H)	"We also found, in agreement with [2], [5], [7], [19], [25] that most faults are found in about 20% of the components". [2] is the original study here.
	[Hamill and Goseva-Popstojanova 2015b]	No (H)	"Results are,..., not consistent with others (Fenton and Ohlsson, 2000; Ostrand and Weyuker, 2002)"
[Menzies et al. 2007]	[Turhan and Bener 2007]	Yes (G)	"our replicated results are similar to reported mean results in [10],..., Previous research reported mean (pd, pf) = (71,25) which yields bal = 72 averaged over all datasets. Replication of these experiments yield mean (pd, pf) = (64, 19) and bal = 71". [10] is the original study here.
	[Zhang et al. 2007]	Yes (G)	"This study confirms that static code complexity measures can be useful indicators of component quality."
	[Lessmann et al. 2008]	Yes (G)	"Most classifiers achieve promising AUC results of 0.7 and more, i.e., rank deficient modules higher than accurate ones with probability > 70 percent. Overall, this level of accuracy confirms Menzies et al.'s conclusion that "defect predictors are demonstrably useful"... "
	[Song et al. 2011]	Yes (A)	"The mean prediction balance of the MGF framework over the 17 data sets is 66.5 percent, and the mean prediction balance of the proposed framework is 68.2 percent"
		No (H)	"We have compared the proposed framework with MGF's study [23] and pointed out the potential bias in their baseline experiment"
	[Singh and Verma 2014]	Yes (H)	"we also found that data mining using static code attributes to learn fault predictors is useful"
[Moser et al. 2008]	[Krishnan et al. 2013]	Yes (H)	"The previous study by Moser et al. [7],..., did not mention using any statistical test to check for prominence, we find that there is some overlap between those results and our results"
[Kim et al. 2007]	[Rahman et al. 2011]	Yes (H)	"This paper mounts a replicated study of FixCache and,..., does indeed find more bug-dense entities."
[Zimmermann and Nagappan 2008]	[Tosun et al.]	Yes (H)	"Both replication experiments and our results reveal that network metrics are significant indicators of defective modules in large and complex systems"
	[Nguyen et al. 2010]	Yes (H)	"We confirm prior results on an industrial system using an open source system. SNA metrics do improve the prediction performance for quality models."
	[Premraj and Herzig 2011]	Yes (H)	"we replicate their study on three open source Java projects, viz., JRuby, ArgoUML, and Eclipse. Our results are in agreement with the original study by Zimmermann and Nagappan when using a similar experimental setup as them (random sampling)."
[Amasaki et al. 2003]	[Okutan and Yıldız 2014]	Unknown (H)	—
[Schröter et al. 2006]	[Duala-Ekoko and Robillard 2009]	Yes (H)	"We compared the data reported by Schröter et al. to our replication study, and all the imported types reported to be risky by Schröter et al. were also risky in our data."
[Zimmermann et al. 2007]	[Kpodjedo et al. 2011]	Unknown (H)	—
[Khoshgoftaar and Seliya 2002]	[Li et al. 2005]	Yes (H)	From the original study the model results (3 regression models) measured in average absolute error (AAE) followed by average relative error (ARE) are: cart-lad (1.1308,0.3943), s-plus (1.2889, 0.6845), cart-ls (1.2894, 0.6853). The replication results are: Trees Split with 2 Releases (1.4 1.2), Tree Split with 4 Releases (1.4 1.3). Note that it is not clear which of the 3 regression models is replicated from the original study. The replication has similar conclusion with the original that "simple approaches can produce accurate predictions for mature software development organizations"