Researcher Bias: The Use of Machine Learning in Software Defect Prediction

Martin Shepperd, David Bowes and Tracy Hall

Abstract—
Background. The ability to predict defect-prone software components would be valuable. Consequently, there have been many empirical studies to evaluate the performance of different techniques endeavouring to accomplish this effectively. However no one technique dominates and so designing a reliable defect prediction model remains problematic.
Objective. We seek to make sense of the many conflicting experimental results and understand which factors have the largest effect on predictive performance.
Method. We conduct a meta-analysis of all relevant, high quality primary studies of defect prediction to determine what factors influence predictive performance. This is based on 42 primary studies that satisfy our inclusion criteria that collectively report 600 sets of empirical prediction results. By reverse engineering a common response variable we build a random effects ANOVA model to examine the relative contribution of four model building factors (classifier, data set, input metrics and researcher group) to model prediction performance.
Results. Surprisingly we find that the choice of classifier has little impact upon performance (1.3%) and in contrast the major (31%) explanatory factor is the researcher group. It matters more who does the work than what is done.
Conclusion. To overcome this high level of researcher bias, defect prediction researchers should (i) conduct blind analysis, (ii) improve reporting protocols and (iii) conduct more intergroup studies in order to alleviate expertise issues. Lastly, research is required to determine whether this bias is prevalent in other applications domains.

Index Terms—Software defect prediction, meta-analysis, researcher bias.

---

1 INTRODUCTION

Being able to predict defect-prone software modules is an important goal in software engineering. This would enable efficient allocation of testing resources and better informed decisions concerning release quality. Consequently there have been many defect prediction studies. For example a 2009 mapping study by Catal and Diri [1] identified 74 such studies; subsequently the number has grown rapidly. The recent systematic review by Hall et al. [2] located no less than 208 primary studies and showed that predictive performance varied significantly between studies. The meta-analysis contained within this paper is based on the primary studies identified by Hall et al. [2] and explores what causes predictive performance to vary so much between them.

The literature makes two things clear about defect prediction. First, no single prediction technique dominates [3] and, second, making sense of the many prediction results is hampered by the use of different data sets, data pre-processing, validation schemes and performance statistics [4], [3], [5], [6]. These differences are compounded by the lack of any agreed reporting protocols or even the need to share code and algorithms

- *Martin Shepperd and Tracy Hall are with Brunel University, Uxbridge, Middlesex, UB8 3PH,UK.*
  E-mail: {martin.shepperd,tracy.hall}@brunel.ac.uk
- *David Bowes is with Science and Technology Research Institute, University of Hertfordshire, Hatfield, Hertfordshire, AL10 9AB, UK.*
  E-mail: d.h.bowes@herts.ac.uk

[7]. This means that it is difficult for researchers, and more importantly practitioners, to have any clear sense of which defect prediction techniques to employ or how well they are likely to perform.

In order to address these problems and form some overall view, we perform a meta-analysis (that is pooling the results) of *all* relevant primary studies that satisfy our inclusion criteria and in particular provide sufficient details to enable quantitative analysis. In this way we construct the most comprehensive picture of current research possible. However, in order to have a common basis for comparison we require a common performance statistic (or response variable) from all the individual primary studies or experiments. Unfortunately there is no uniformity amongst researchers as to how classification performance should be reported. This lack of uniformity has necessitated us reverse engineering a common performance statistic (the Matthews correlation coefficient (MCC) [8]) for the 600 sets of prediction results published in the 42 primary studies we analyse. The details and rationale of our reverse engineering process are discussed in some detail in the Methods section of this paper.

We then perform a meta-anlysis using an ANOVA model to explain the sources of variance in the classification performance between the various individual studies. Given that the purpose of these primary studies is chiefly to compare different approaches and algorithms for prediction one might expect the bulk of that variability to be associated with the choice of prediction

approach. Importantly, we find this not to be the case; instead much of the variability in performance from different studies originates from other sources, predominantly who does the research. It would therefore seem unlikely that much progress will be made in building practically useful defect prediction models — despite the considerable efforts of many research teams — until we better understand and control for these unwelcome sources of variability.

Although the focus of this meta-analysis is software defect prediction, and specifically binary classifiers that assign software modules to one of two classes (faulty or non-faulty), many of the principles of this analysis can be applied far more widely. In particular we believe the presence of unintentional researcher bias and the need to address this make the results we report of some considerable significance to other communities both in software engineering and beyond.

The remainder of this paper is organised as follows. The next section reviews current software defect prediction research followed by a discussion of using meta-analysis for the results from multiple primary studies. Section 3 describes issues with the measurement of classification performance, how the candidate primary studies were selected and how we extracted a common performance response variable from published results. Then, we describe the meta-analysis itself along with associated results. Section 5 gives a discussion of the implications of our findings, a brief review of scientific bias in other disciplines and concludes with three specific recommendations as to how bias might be reduced for future research into empirically evaluating defect prediction systems. The Supplementary Material contains an exploration of some alternative approaches.

2 RELATED WORK

2.1 Software Defect Prediction

For some time software engineers have been interested in predicting the quality of software systems that are being developed. A specific focus is to predict software defects and where they may be located, for instance which modules are likely to be particularly defect-prone. In order to achieve this, researchers have explored many different techniques that can generate a prediction system from known training examples. Since it is unclear *a priori* which techniques are 'best' since we lack any deep theory, experimental validation is required.

The general philosophy of experimental validation of a putative prediction system is as follows. A prediction system — and since we are exclusively dealing with binary prediction we term them classifiers — is induced by some means over some training instances which have known labels or outcomes. This is known as supervised learning. Because there is a danger of over-fitting to the training instances the classifier must be evaluated on *unseen* instances, usually referred to as testing instances. The question of how a particular classifier can be trained

and then cope with new data is an empirical question. Consequently, the vast majority of studies predicting defect-prone modules use either primary, or more often, secondary defect data that has been collected from real-world software projects.

Ideally, empirical validation of classifiers would use completely new data for testing purposes to reflect how we might expect them to be deployed in practice. Unfortunately the reality of how researchers work is that data are in short supply and so we wish both to use what data we have effectively and to realistically simulate the deployment process. This is achieved through cross-validation [9]. There are a range of strategies but essentially they all partition the data into training and test instances and this is done randomly and repeatedly in order to maximise the potential of limited data and minimise the likelihood of outliers unduly influencing the results. The results of the repetitions are then combined typically using some measure of central tendency (see [10] for a more detailed discussion). The most common approach is $m \times n$-fold cross validation.

As we have indicated, software defect prediction has attracted many investigators over the past two decades encouraged by the potential cost and quality rewards associated with accurately predicting defect-prone software [1], [2]. There is no expectation that a single, universal classifier would be of any practical value so researchers have concentrated on techniques for learning or building classifiers from local or relevant data. These studies vary greatly, but all have four basic elements:

1) the learning technique by which the particular classifier is derived
2) the defect data used to build and test the classifier
3) the types of input metrics or features used
4) the predictive performance measures used.

We will briefly discuss each in turn.

A wide variety of classifier building techniques have been used by researchers. The dominant approach for software defects is categorical prediction, usually dichotomous (and only so in this study) so that the two classes are defect-prone and not defect-prone software components e.g., classes, files or subsystems. Such approaches can be characterised into more specific techniques including rule induction algorithms such as C4.5, Neural Networks, Random Forests, Bayesian approaches and Regression which for binary classification is Logistic Regression. The majority of studies have compared the performance of classifiers using multiple learning techniques and sometimes, though frequently not, with some benchmark technique such as guessing with the odds following the known distribution of classes.

Second, in order to assess classifier building techniques most researchers have used real world data though frequently the data is secondary, that is it has been previously collected by other researchers. There are various mechanisms to facilitate this sharing of data, with the Promise Data Repository [11] being at the forefront of such initiatives. Whilst sharing of data is

clearly a good thing it is not without risk, particularly when problems and errors in the data are propagated [12], [13]. However, it does afford us the opportunity to examine the impact of other factors upon defect prediction performance since many different researcher groups have used the same data.

In addition, there is a debate on whether classifiers are transferable to new settings or whether classifiers are context specific [14]. But in any case, if only for pragmatic reasons of exactly which data features are available in a particular setting, it is necessary to be able to learn new classifiers and therefore generalisation is through the learning technique rather than via any specific classifier. Consequently, the primary research preoccupation is which classifier learning techniques work best, perhaps with the rider of in what circumstances.

Third, a wide range of metrics have been used as inputs to defect prediction models. These are divided into three classes by Arisholm et al. [15]: structural measures, delta (or change) measures and process measures. Structural measures are based on aspects of the code such as size and complexity measures (example studies include [S31]). Delta measures relate to the change that has taken place to the code, for example how the structure of code has changed over versions of a code unit (example studies include [S21]). Process measures focus on how the code has been developed and include developer experience (example studies include [S2]).

Fourth, there is no consensus on how to measure classification performance. It is well known that simple accuracy measures are misleading in the face of imbalanced data sets [16]. By way of illustration consider the situation where 95% of software components are fault free and only 5% contain defects, then the naïve strategy of predicting all modules are the modal class, that is fault-free will achieve a 95% accuracy rate. However this would hardly constitute a useful prediction system!Note that such distributions are relatively commonplace amongst the defect data sets used by researchers (see Figure 1 which reveals that over 40% of data sets used by the 42 studies we analyse have less than 20% instances of the positive class i.e. defect-prone). For this reason the majority of researchers report either the F-measure [17] or Area under the Curve (AUC) [18], [16]. We will discuss in more depth some of the subtleties of these choices in Section 3.2, however, a wide range of different statistics have been used and this has complicated meaningful comparison between different primary studies [19].

To summarise, there has been very substantial research effort put into software defect prediction. However, a consensus on what are the best prediction techniques, even for a specific context, remains elusive. We find considerable diversity as to what form of classifier technique should be used and what inputs or metrics work best. In addition, in order to facilitate generalisation, researchers are using an increasing number of different defect data sets for empirical validation. Unfortunately the situation
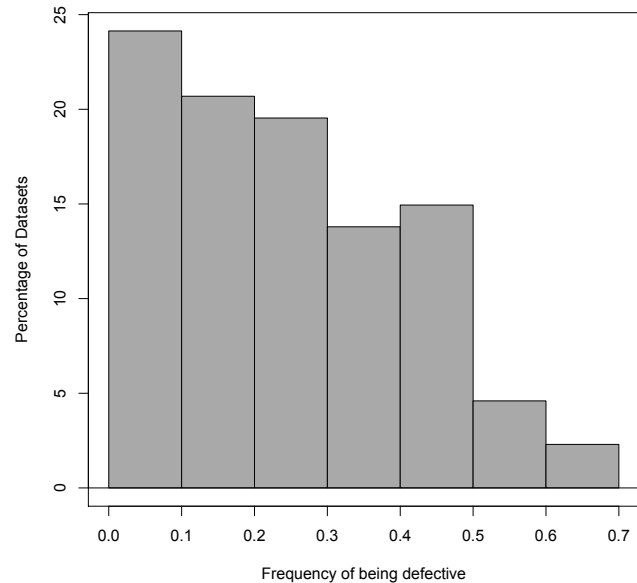

Figure 1. Percentage Distribution of Positive Case Imbalance in Data Sets

is somewhat complicated by the use of a wide range of validation schemes. We believe the time is ripe to explore the underlying reasons for this lack of convergence in results hence we conduct a meta-analysis. This will provide pointers to the most effective way forward for future software defect prediction research.

2.2 Meta-analysis

Meta-analysis is some procedure for aggregating and analysing results from more than one primary study. They are often, but not necessarily, a part of a systematic literature review which has the broader remit to also include identification of the primary studies in some objective and repeatable fashion driven by a formal, and ideally public, protocol (see [20] for details). Normally meta-analysis takes the form of a quantitative statistical procedure. This is clearly more straightforward when the primary studies yield quantitative results (i.e., predictive performance) as is the case with our defect prediction studies. The aggregation approach can either utilise results or pool the raw data. In our case the latter is not possible due to difficulties in obtaining all the raw results. Alternatively results can be pooled, though with the caveat that simplest approach of using vote counting is vulnerable to heterogeneity amongst the primary studies, particularly in terms of size or quality. For this reason stringent quality inclusion criteria are required, coupled with sensitivity analysis of the meta-analysis results [21].

There is some disagreement as to what was the first meta-analysis, however the term was coined by Glass in the mid 1970s [22]. Meta-analysis has only gained

interest within empirical software engineering relatively recently, despite initial work in the late 1990s [23], [24].

Naturally synthesising findings across studies presents challenges. The Cruzes and Dybå [25] analysis of 31 systematic literature reviews in software engineering found that limited attention was paid to the actual synthesis of results as opposed to merely cataloguing what existed. Less than half of all reviews contained any synthesis and of those that did a narrative or relatively informal approach predominated. Only two used meta-analysis as their synthesising technique. Even in more mature disciplines where meta-analysis is commonplace, e.g. medicine, there can be challenges.

Thus the general view is that meta-analysis is a powerful tool but, unsurprisingly, can be abused when:

- it is used with no underlying model or theory. A meta-analysis example from epidemiology was highlighted by Egger et al. [26] that suggested causality between smoking and suicide rates which is biologically implausible, rather it is the social and mental states predisposing to suicide that also co-vary with smoking. Nevertheless it can be easy to persuade oneself of the 'plausibility' of some set of results *post hoc*.
- meta-analysis cannot turn poor quality studies into 'gold'. If a primary study is of very low quality then its conclusions are jeopardised irrespective of how it is merged with other studies [27].
- sources of heterogeneity are ignored e.g. the variance between types of study, use of blinding, etc. though this can be addressed through sensitivity analysis, especially based on mixed or random effects models [28].

Despite these cautions meta-analysis offers a powerful and formal means of quantitatively aggregating results from multiple studies. This can help overcome chance outcomes (since even unlikely events can and do occur!), can provide better sampling and highlight weak experimental procedures. Therefore we propose to conduct a formal meta-analysis of all relevant, high quality empirical studies into software defect prediction.

3 METHOD

3.1 Selection of the Primary Studies

Our aim is to be as comprehensive as possible in terms of selecting primary studies that are experimental studies of software defect prediction models and further, these must predict a dichotomous outcome (i.e, defect or not defect-prone). This focus is essential in order to compare like with like primary studies. Inclusion, requires (i) satisfying various quality criteria so that the meta-analysis is not distorted by methodologically weaker primary studies and (ii) that sufficient details are provided such that we can extract, or reverse engineer, our chosen response variable so that we can meaningfully compare across these primary studies.

A previous systematic literature review of defect prediction studies in software engineering [2] identified a total of 208 studies published from the start of 2000 to the end of 2010. A set of criteria to assess whether or not to extract data from prediction models presented in these 208 studies is developed in [2] and summarised in Table 1. These criteria are based on the context reporting standards recommended by Petersen and Wohlin [29] and on best practice reported in the defect prediction literature (e.g. [6]). They require that a study reports a set of basic information that enables full understanding of the prediction models presented.

Table 1
Summary of Primary Study Inclusion Criteria (Taken from [2])

Topic
1. Published between January 2000 and December 2010
2. An empirical study / experiment
3. Focused on predicting faults in units of a software system
4. Faults in code is the main output (dependent variable)
5. Not testing, fault injection, inspections, reliability modelling, aspects, effort estimation, debugging, faults relating to memory leakage, nano-computing, fault tolerance
6. Not debugging of known faults
Quality and Reporting Considerations
7. The classifier must be trained and tested on different data
8. Provide information on:
- source of data
- the maturity of the system being studied must be reported (relaxed for NASA)
- the size of the system being studied must be given
- the application domain
- programming language(s)
- the independent variables
- the dependent variable(s)
- the granularity of the dependent variable reported
- the modelling [classifier learning] technique used reported
- the fault data acquisition process
- the independent variable data acquisition process
- the number of faulty versus non-faulty units

In this meta-analysis we started with 23 studies which passed the criteria in [2] plus a further 24 studies which were originally excluded from [2] because they did not provide information on the maturity of the data sets used (though they satisfied all other criteria). All 24 of these studies are based on using NASA data. These were added since we are able to obtain additional information concerning the NASA data sets from other sources and although concerns have been raised about data quality this is in part due to the fact that they have been more thoroughly scrutinised than other data sets [12], [13]. Consequently we believe them to be typical of much data used for this kind of experimentation. Moreover our goal is to try to better understand why results vary as opposed to uncovering which classification techniques are 'best'.

From this set of 47 prediction studies, we exclude a further five studies which do not allow us to compute the base confusion matrix measures (see Table 3), without which we cannot extract a prediction quality response variable and so there is no common response variable.

<div style="text-align: center">

Table 2
Included Primary Studies

</div>

Primary Study Reference	Count	Year
[S16], [S20], [S27]	3	2002
[S26], [S34]	2	2004
[S15], [S25], [S31]	3	2005
[S14], [S33], [S38]	3	2006
[S3], [S4], [S6], [S10], [S13], [S21], [S29], [S36], [S37], [S41]	10	2007
[S8], [S9], [S17], [S30], [S35], [S39]	6	2008
[S1], [S2], [S5], [S7], [S11], [S40], [S42]	7	2009
[S12], [S18], [S19], [S22], [S23], [S24], [S28], [S32]	8	2010

<div style="text-align: center">

Table 3
Confusion Matrix

</div>

	Predicted defective	Predicted defect free
Observed defective	True Positive (TP)	False Negative (FN)
Observed defect free	False Positive (FP)	True Negative (TN)

<div style="text-align: center">

The confusion matrix forms the fundamental basis from which
almost all other performance statistics can be derived.

</div>

This leaves us with a total of 42 primary studies (listed in Table 2) that (i) report sufficient performance information for us to reconstruct their confusion matrices with confidence and (ii) satisfy our quality inclusion criteria.

3.2 Prediction Performance Measures

Assessing the prediction performance of a classifier is based upon the confusion matrix (see Table 3 where the cells comprise frequencies for each combination of the two binary dichotomous variables. In general the meanings of the values of the binary variables need not be defined, however, for assessing classification performance we are more specific. The class labels are termed positive and negative. It is customary to use the positive label to refer to the outcome of interest so in our situation the positive label refers to the software component being defective. Hence we have true-positives (TP , that is the instances of defects that are correctly classified, false-positives (FP) which are those instances that the classifier incorrectly assigns to the defect-prone class and so forth.

Determining classification performance is more subtle than it might first appear since we need to take into account both chance aspects of a classifier (even guessing can lead to some correct classifications) and also what are termed unbalanced data sets where the distribution of classes is far from 50:50. As discussed previously this is a commonplace situation for defect data sets since most software units do not contain known defects.

Many commonly used measures such as the F-measure are unsatisfactory due to their not being based on the complete confusion matrix [16]. A widely used alternative is the Area Under the Curve (AUC) ROC chart [30], however, this because this is a measure on a family of classifiers it cannot be interpreted unless one classifier

strictly dominates since since we're not provided with the relative costs of FP and FN. In other words for two classifiers A and B, $AUC_A > AUC_B \not\Rightarrow A \succ B$. For this reason we advocate a binary correlation coefficient variously known as the Matthews correlation coefficient (MCC) [8] or ϕ [31]. Unlike the F-measure, MCC is based on all four quadrants of the confusion matrix.

MCC is identical to the binary correlation coefficient ϕ originally due to Yule and Pearson in the early twentieth century [31]. However, we will refer to the measure as MCC rather than ϕ since we are dealing with the particular binary variables and meanings of predicted and actual class whereas the ϕ coefficient is a more general measure of association, where the interpretation of a negative correlation is more difficult to resolve. It is a balanced measure and handles situations where the ratio of class sizes are highly imbalanced which is typical of software defect data (classes containing defects are often relatively rare). A zero value indicates the two variables are independent, tending towards unity indicates a positive relationship between predicted and actual class and tending towards minus unity a negative relationship. Interestingly one could convert a very poor or perverse classifier into a very good one by negating the prediction (turning a positive to negative or vice versa). Finally, MCC can be easily computed from the confusion matrix.

One slightly awkward property of the ϕ coefficient, and therefore MCC, is that depending upon the marginal distributions of the confusion matrix, plus or minus unity may not be attainable and so some statisticians propose a ϕ/ϕ_{max} rescaling [32]. We choose not to follow this procedure since it results in an over-sensitive measures of association in that a very small change in the count of correctly classified instances (TP or TN) lead to unintuitively large changes in the correlation. We discuss this in more detail together with some potential limitations of using a correlation coefficient in a meta-analysis in the Supplementary Materials. Nevertheless, we believe it is a good choice to evaluate classifier performance in a meta analysis of published studies.

The 42 primary studies used for the meta-analysis between them comprise 600 individual results from which we extracted performance data i.e., MCC. Typically each study reported one or more experiments where a number of treatments such as different classifier learning techniques are compared using one or more data sets. This is usually coupled with statistical inferencing procedures to determine whether the differences in the response variable are in some sense significant. However our concern is simply with the raw results. Where studies did not report MCC we re-computed it from the confusion matrix-based data that they provided ([33] explains our method for doing this). Identifying a common set of performance measures for these 42 studies enables a quantitative meta-analysis of the performance of all the reported 600 empirical results.

Table 4
Composite performance measures

Construct	Defined as	Description
Recall pd (probability of detection) Sensitivity True positive rate	$TP/(TP+FN)$	Proportion of defective units correctly classified
Precision	$TP/(TP+FP)$	Proportion of units correctly predicted as defective
pf (probability of false alarm) False positive rate	$FP/(FP+TN)$	Proportion of non-defective units incorrectly classified
Specificity True negative rate	$TN/(TN+FP)$	Proportion of correctly classified non defective units
F-measure	$\frac{2 \cdot Recall \cdot Precision}{Recall+Precision}$	Most commonly defined as the harmonic mean of precision and recall
Accuracy	$\frac{(TN+TP)}{(TN+FN+FP+TP)}$	Proportion of correctly classified units
Matthews Correlation Coefficient	$\frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$	Combines all quadrants of the binary confusion matrix to produce a value in the range -1 to +1 with 0 indicating independence between the prediction and the recorded results. MCC can be tested for statistical significance, with $\chi^2 = N \cdot MCC^2$ where N is the total number of instances.

3.3 Meta-Data Extraction

In order to explore the factors that influence experimental results we need to consider those factors that influence each primary experiment. These are known as moderators. We use the following:

1) learning technique by which the particular classifier is derived
2) defect data set used to build and test the classifier
3) types of input metrics or features used
4) research group

Note we do not need to consider the original choice of predictive performance since we recompute a common response variable for all experiments, namely MCC.

Having determined which factors or moderators to use, two authors (TH and MS) each independently read and categorised each paper within the meta-analysis. The third author (DB) identified any inconsistencies which were then resolved through discussion between all three authors. Consequently we believe the extraction process to be quite robust.

3.3.1 Classifier family

We grouped specific prediction techniques into a total of seven families which were derived through a bottom-up analysis of each primary study. The categories are given in Table 9. This avoided the problem of every classifier being unique due to subtle variations in parameter settings, etc.

3.3.2 Data set family

Again these were grouped by base data set into 24 families representing 87 distinct defect data sets that have been used in these studies. These groupings meant that differing versions of systems were not regarded

as entirely new data sets. There are also substantial overlaps in data set usage by primary studies including many based, at least in part, on the NASA data sets or on the Eclipse data set.

3.3.3 Metric family

Again we group specific metrics into families so for example, we can explore the impact of using change or static metrics. We use the classification proposed by Arisholm et al. [15], namely Delta (i.e. change metrics), Process metrics (e.g. effort) and Static metrics (i.e. derived from static analysis of the source code e.g. the Chidamber and Kemerer metrics suite [34]) and then an Other category(e.g. the use of metrics derived from socio-technical networks [S11]). Combinations of these categories are also permitted (see Table 10). Again our philosophy is to consider whether gross differences are important rather than to compare detailed differences in how a specific metric is formulated.

3.3.4 Researcher Group

The researcher groups were determined by linking researchers through co-authorship to form clusters of researchers. The largest clusters (of 8 and 10 researchers) derived from three papers apiece. The smallest cluster comprises a single author. This is depicted by Figure 2 which provides a visual though arbitrary impression of the diversity of research groups. The cluster numbering is arbitrary but provides a key to Table 5 which details each group (note the first instance of each author is in bold).

4 RESULTS

We use R (open source statistical software) for our analysis.

Table 5
Researcher Groups

| Group	Paper Ref	Researchers
1	[S1]	**Calikli, G** and **Tosun, A** and **Bener, A** and **Celik, M**
	[S2]	**Caglayan, B** and Bener, A and **Koch, S**
	[S3]	**Kutlubay, O** and **Turhan, B** and Bener, A
	[S4]	Turhan, B and Bener, A
	[S5]	Tosun, A and Bener, A
2	[S6]	**Menzies, T** and **Greenwald, J** and **Frank, A**
3	[S7]	**Kaur, A** and **Sandhu, P** and **Bra, A**
	[S8]	Kaur, A and **Malhotra, R**
	[S9]	**Singh, Y** and Kaur, A and Malhotra, R
4	[S10]	**Pai, G** and **Dugan, J**
5	[S11]	**Bird, C** and **Nagappan, N** and **Gall, H** and **Murphy, B** and **Devanbu, P**
	[S12]	Nagappan, N and **Zeller, A** and **Zimmermann, T** and **Herzig, K** and Murphy, B
	[S13]	Zimmermann, T and **Premraj, R** and Zeller, A
	[S14]	**Schröter, A** and Zimmermann, T and Zeller, A
6	[S15]	**Koru, A** and **Liu, H**
7	[S16]	**Denaro, G** and **Pezzè, M**
8	[S17]	**de Carvalho, A** and **Pozo, A** and **Vergilio, S** and **Lenz, A**
	[S18]	de Carvalho, A and Pozo, A and Vergilio, S
9	[S19]	**Nugroho, A** and **Chaudron, M** and **Arisholm, E**
	[S20]	**Briand, L** and **Melo, W** and **Wust, J**
	[S21]	Arisholm, E and Briand, L and **Fuglerud, M**
	[S22]	Arisholm, E and Briand, L and **Johannessen, E**
10	[S23]	**Cruz, C** and **Erika, A**
11	[S24]	**Yi, L** and **Khoshgoftaar, T** and **Seliya, N**
	[S25]	Seliya, N and Khoshgoftaar, T and **Zhong, S**
	[S26]	Khoshgoftaar, T and Seliya, N
	[S27]	Khoshgoftaar, T and **Yuan, X** and **Allen, E** and **Jones, W** and **Hudepohl, J**
12	[S28]	**Mende, T** and **Koschke, R**
13	[S29]	**Catal, C** and **Diri, B** and **Ozumut, B**
14	[S30]	**Moser, R** and **Pedrycz, W** and **Succi, G**
15	[S31]	**Gyimothy, T** and **Ferenc, R** and **Siket, I**
16	[S32]	**Vivanco, R** and **Kamei, Y** and **Monden, A** and **Matsumoto, K** and **Jin, D**
17	[S33]	**Ma, Y** and **Guo, L** and **Cukic, B**
	[S34]	Guo, L and Ma, Y and Cukic, B and **Harshinder Singh**
	[S35]	**Yue Jiang** and Cukic, B and Ma, Y
18	[S36]	**Mizuno, O** and **Ikami, S** and **Nakaichi, S** and **Kikuno, T**
	[S37]	Mizuno, O and Kikuno, T
19	[S38]	**Yuming Zhou** and **Hareton Leung**
20	[S39]	**Vandecruys, O** and **Martens, D** and **Baesens, B** and **Mues, C** and **De Backer, M** and **Haesen, R**
21	[S40]	**Hongyu, Z**
22	[S41]	**Kanmani, S** and **Uthariaraj, V** and **Sankaranarayanan, V** and **Thambidurai, P**
23	[S42]	**Shivaji, S** and **Whitehead, E** and **Akella, R** and **Sunghun, K**
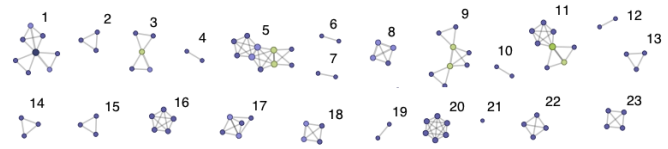
Figure 2. Researcher Collaborations

4.1 Descriptive Analysis of the Empirical Studies

The variables collected are described in Table 6. They are then summarised in Tables 7 and 8. There are a total of seven Classifier family categories and the relative distributions are shown in Table 9. It is clear that Decision Trees, Regression-based and Bayesian approaches predominate with 172, 136 and 124 observations respectively.

Table 6
Variables Used in the Meta-analysis

Variable	Explanation
MCC	Matthews correlation coefficient also known as ϕ
Year	Date of study publication
ClassifierFamily	The class of classifier e.g. regression, case-based reasoner, etc.
DatasetFamily	The data set family (excludes version number) used for validation e.g. Eclipse and Mozilla.
MetricFamily	The category of metric used as input to the classifier.
ResearcherGroup	The cluster of researchers where one or more common researchers between papers creates a linkage and therefore a single group.

Table 7
Summary Statistics for the Numeric Variables

Variable	Min	Median	Mean	Max
MCC	-0.504	0.305	0.308	0.989
Year	2002	n.a.	n.a.	2010

Table 8
Summary Statistics for the Categorical Variables

Variable	No. of classes
ClassifierFamily	8
DataSetFamily	24
MetricFamily	7
ResearcherGroup	23

Second, we consider Dataset family. Here there are 24 different classes but NASA dominates with approximately 59% of the observations followed by Eclipse with 21% (see Table 11). We believe a major factor determining choice of data set is easy availability which could partly explain why many researcher groups use the same data. On the other hand one needs to be sure that the data are representative and that any problems with data quality are not propagated [12].

Third, is the Metric family. We observed a total of seven distinct classes of metric category used as inputs for the classifiers. The relative distributions are shown in Table 10. It can be seen that Static metrics, that is those extracted through static code analysis, dominate being used in almost 85% of all studies.

Table 9
Frequency of Instances by Classifier Family

	Instances	Percent
DecTree	172	28.7
Regression	136	22.7
Bayes	124	20.7
CBR	77	12.8
Search	41	6.8
ANN	28	4.7
SVM	17	2.8
Benchmark	5	0.8

Table 10
Frequency of Instances by Metric Family

	Instances	Percent
Static	507	84.5
Other	28	4.7
DeltaProcessStatic	26	4.3
Process	16	2.7
DeltaOtherProcessStatic	11	1.8
DeltaStatic	11	1.8
ProcessStatic	1	0.2

Table 11
Result Count by Data Set Family

	Count	Percent
NASA	351	58.5
ECLIP	126	21.0
EMTEL	25	4.2
MOZ	25	4.2
COS	16	2.7
LibMan	12	2.0
EXCL	9	1.5
WhiteGoods	6	1.0
VISTA	4	0.7
APACHE	3	0.5
ARGO	3	0.5
HEALTH	3	0.5
BANK	2	0.3
CruiseControl	2	0.3
Ecomerce	2	0.3
JEDIT	2	0.3
XPOSE	2	0.3
COLUMBA	1	0.2
GALM	1	0.2
GFORGE	1	0.2
JCP	1	0.2
PONE	1	0.2
POSTGRES	1	0.2
SVN	1	0.2

Fourth, is researcher group determined by clustering based on common researcher linkage so if A and B write Paper 1 and B and C write Paper 2, this generates a single cluster of A, B and C. The reasoning is that expertise will be shared amongst the entire team. As Figure 2 reveals, from our 42 primary studies there are 23 relatively small

Table 12
Frequency of Instances by Researcher Group

	Result Count	Percent
17	79	13.2
1	75	12.5
8	61	10.2
11	54	9.0
5	53	8.8
12	37	6.2
20	31	5.2
9	30	5.0
13	30	5.0
14	27	4.5
15	24	4.0
16	18	3.0
21	15	2.5
22	12	2.0
23	11	1.8
18	10	1.7
19	9	1.5
2	8	1.3
10	6	1.0
3	5	0.8
4	2	0.3
7	2	0.3
6	1	0.2

clusters ranging in size between 1 and 10 researchers. The number of results that each group report is shown in Table 12.

In particular we are interested in the distribution of the response variable MCC. Figure 3 shows that the distribution is approximately symmetrical. Likewise the qqplot in Figure 4 is near to linear suggesting a reasonable approximation to a normal distribution, though with some deviation at the tails. This deviation at the tails is commonplace for correlation coefficients [35] as they are bounded by unity. The standard correction is the Fisher r to Z transformation, however as discussed in more detail in the Supplementary Material this turns out to be an unhelpful procedure for our meta-analysis with the resultant distribution deviating further from normality. We believe the most likely explanation to be twofold. First, our observations are assuredly not a random sample since researchers tend to report their 'best' or most 'interesting' results. Second, depending upon the marginal probabilities of the confusion matrix MCC is constrained to values less than unity (minus unity) [32]

What is striking, however, is the number of observations of the correlation coefficient being close to zero or even negative. This reveals that many classifiers are performing extremely poorly, since zero indicates no relationship at all and could therefore be achieved through guessing. A negative value means the prediction would be improved by adding an inverter. Moreover, the modal value lies in the range 0.3-0.4 which hardly engenders a great deal of confidence in terms of their practical use.

Next we check for possible autocorrelation effects since the primary studies were published between 2002 and 2010 (see Table 2). In other words are there any maturation effects over time (e.g. researchers have become more expert and therefore more accurate). Fig-
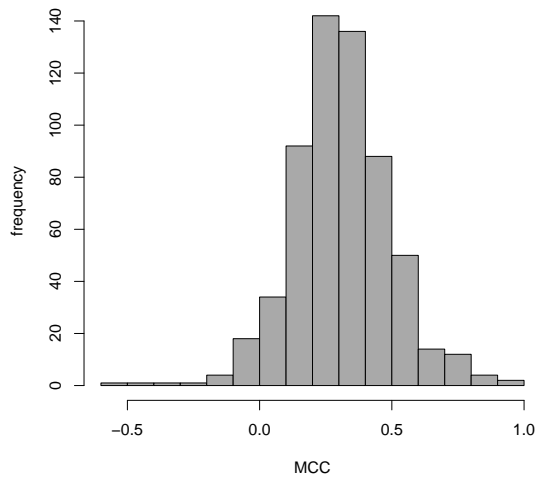
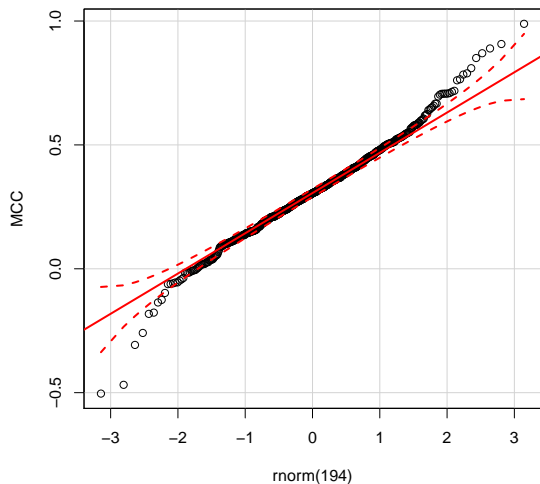Figure 3. Histogram of Prediction Performance as Matthews Correlation Coefficients

Figure 5. Matthews Correlation Coefficients by Year

Figure 4. qqplot of the Matthews Correlation Coefficients

ure 5 shows the relationship between year (with jittering) and MCC. The slight upward trend indicated by a linear model is not significant and the R-squared is approximately 1% (we have previously reported a lack of methodological maturation in defect prediction studies [36]). This means that we can safely discount autocorrelation as a potential confounder and can now proceed to our main analysis based on MCC as the response variable and Classifier Family, Data Set Family, Metric Family and Researcher Group as moderators plus potential interactions between moderators.

4.2 ANOVA Model Results

We use a 4-way random effects ANOVA model where MCC is the response variable and the four factors (or
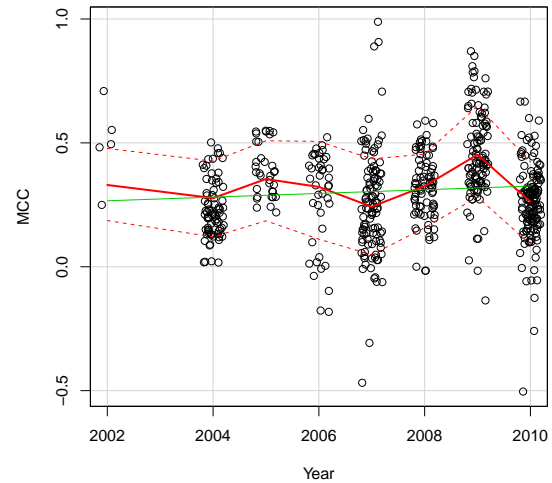
moderators) are:

- Classifier family
- Data set family
- Metric family
- Researcher group

Since this is not an orthogonal factorial model the order the factors are entered into the model is significant as we will discuss shortly. In addition we are also interested in higher order interactions, e.g. the interaction between Classifier Family and Dataset Family. The rationale for this is that there may be a significant interaction between the dataset and predictive performance. This has been demonstrated elsewhere through simulation studies [37].

In this meta-analysis we wish to see how the four factors contribute to the variation in performance as measured by Matthews correlation coefficient. Or to put the question rather more crudely: what matters in terms of performance? An important point to stress is we use random effects models. That is, we are interested in the factor but not the specific levels, e.g. we are interested in how much influence the choice of data set has on predictive performance but not whether using the Eclipse data set, say, leads to greater or poorer performance than Mozilla.

To reiterate, we are *not* seeking to determine which is the 'best' technique for learning classifiers to predict software defects but why results from different primary study experiments vary. Our approach yields two benefits. First, it provides more generality since we can reason about hitherto unseen data sets, classifiers and so forth. Second, the approach is more robust to unequal distribution of variance[1] between levels within a factor

---

[1]Testing for equality of variance in an ANOVA model is not necessarily straightforward [38]. Complications include unbalanced levels for different factors some with very few observations. However, a robust Levene's Test for Homogeneity of Variance based on medians yields p=0.51 so we consider it reasonable to proceed.

since we will not test for differences between specific pairs of levels within a factor [39].

First, however, we explore each factor by itself (using a linear random effects model with intercept for the 1-way ANOVA) before proceeding to build more complex models where the factors are entered in decreasing order of partial-eta squared value. That is the most important factor goes first since our factors are not necessarily orthogonal so some variance might conceivably be allocated to one of several factors depending upon the ordering of the factors in the model. As is conventional ([40], [38], [41], [42]) we construct the model in decreasing order of partial eta squared scores (see Table 13). All four factors are significant. Note that a partial-eta squared value indicates the proportion of the total variability in the classifier performance that is associated with the factor when all other factors are excluded determined by a 1-way Analysis of Variance. When a more complete model, i.e., multi-factor is constructed the ANOVA procedure will allocate variance between factors so unless they are orthogonal the actual eta-squared value for a particular model will decrease, excepting the first, and most important, factor in the model.

Table 13
Partial Eta-Squared Values for the ANOVA Model (MCC = Response Variable)

Factor	Partial η^2	Significance
Researcher Group	31.01%	$p < 0.0001$
Dataset Family	31.00%	$p < 0.0001$
Input Metrics	12.44%	$p < 0.0001$
Classifier Family	8.23%	$p < 0.0001$

Starting with the obvious question we look at Classifier Family and its relationship with classifier performance (MCC). Table 13 shows that although Classifier Family is significant it explains approximately 8% of the total variation and the remainder goes to the error term. This is a poor fit and particularly disappointing given the aim of experiments contained in our primary studies are primarily to find the effect of different classifiers upon performance. Instead we see from the analysis of the 600 results that differences are largely due to causes other than the classifier used.

Next, we examine the impact of data set. Elsewhere it has been suggested that different data set characteristics may have a major impact upon prediction system performance [37], [43]. Again we examine a 1-way ANOVA model using Dataset Family to determine the partial eta-squared. Table 13 shows again a significant relationship and an improved fit (compared with Classifier Family) so the choice of data is influential in terms of classifier performance. This would seem to fit with researcher expectations since it is commonplace to apply multiple classifiers to multiple data sets is e.g. [3].

Our third moderator (of experimental results) is the choice of input metric for the classifier. Table 13 shows the results of a 1-way ANOVA where the factor Metric

Family is significant but of limited explanatory value since it covers slightly less than 13% of the total variation in MCC. Interestingly this has not been such a dominant theme to researchers as has been the choice of classifier learning technique, nevertheless, it is associated with 50% more variation in the response variable.

The other moderator to analyse is researcher group which are defined by an agglomerative clustering process and the resultant 23 clusters are shown in Figure 2 and Table 5. Observe that by a small margin Researcher Group has the highest partial eta-squared value indicating that it accounts for the greatest proportion of variance in the response variable of our four factors. Also note that although the difference between it and Dataset Family is small in other analyses of subsets of the results the difference becomes more pronounced [44]. Back in the 1990s Michie et al. [45] commented that that "in many cases researchers have developed their own 'pet' algorithm in which they are expert but less so in other methods" which confounds empirical evaluation studies. It may be that matters have not improved much due to the increasing sophistication and complexity of many machine learning techniques now in use.

So overall we see that the strongest relationship is between Researcher Group and the performance statistic MCC. However, it is likely that some, or all of our four factors interact significantly, and so we move onto a 4-way random effects ANOVA model with interaction terms, e.g. it is possible that the combination of classifier and data set have a particular impact upon performance. We build the model in order of the importance of each factor from the 1-way ANOVA analysis previously described.

Table 14
Eta-Squared Values from the 4-way ANOVA Model (MCC = Response Variable)

	Df	Sum Sq	% of total variance	F value	Pr(>F)
ResearcherGroup	22	6.38	31.0	16.47	0.000
Dataset	20	2.31	11.2	6.55	0.000
ResearcherGroup: Classifier	33	1.36	6.6	2.34	0.000
Metric	5	1.07	5.2	12.19	0.000
Classifier	7	0.26	1.3	2.12	0.040
ResearcherGroup: Dataset	2	0.22	1.0	6.11	0.002
Residuals	510	8.98	43.6		

The model is then simplified by removing factors that are not significant and so we derive Table 14. The resulting model has a good fit (over 55%) when we consider the diversity of primary studies and the relative naïvety of the four moderators. It should also be noted that there remains approximately 45% of external variation (that is the residuals) to our model i.e. the error term equates to less than half of all the variability in the performance statistic MCC. This suggests that defect

prediction studies are unlikely to be very sensitive in detecting *small* differences between competing classifiers since other unknown factors account for just under half the variance in classification performance. What is also striking is that if the factor Researcher Group is removed then the new model accounts for only 35% of the variance, a reduction of 20%. Thus we conclude that knowing who has conducted the research plays an important part in understanding experimental results.

5 DISCUSSION

In this paper we have addressed the question of what factors influence the predictive performance of software defect classifiers measured by the Matthews Correlation Coefficient. The question is important because (i) being able to effectively predict the location of defects would be useful for software engineers and (ii) there is no real congruence between the many empirical results that have been generated such that no classifier technique — amongst the many proposed — dominates.

Fundamentally our finding from the meta-analysis of 600 experimental results drawn from 42 primary studies is that the variability due to the choice of classifier (which has driven much of our research design) is extremely small. Moreover, these are studies that satisfy our rigorous inclusion criteria and have been published in top conferences and journals with demanding acceptance rates. Therefore we conclude that these represent some of our best research efforts to date. By contrast the variability due to the groups of researchers (Researcher Group) could be seen as twenty five times higher[2]. Knowing who has conducted the research means that approximately 30% of total variance in performance can be 'explained'. Until this can be satisfactorily addressed there seems little point in conducting further primary studies since all one can conclude is the main effect is due to the research team undertaking the work.

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. What is rather disappointing is it hardly matters what algorithms or approaches they employ which associates a little over 1% of the variability in the response variable MCC. This finding is *not* the consequence of there being no prediction technique that is uniformly better in all contexts [37]. If a technique worked only in some contexts we would expect the interaction term Classifier:Dataset to be important whilst in practice it isn't significant in our main analysis. We think this is a strength of our meta-analyis in that it allows these kind of second order factors to be explored.

The question therefore arises as to why this might be so. We see two likely and complementary explanations.

[2]The exact ratio depend a little on how the analysis is conducted, however, no model that we examined showed anything other than this to be the strongly dominant factor.

First, there are differing levels of expertise between researcher groups and second, there is the widespread problem of researcher bias. At this juncture we should stress that these are not derogatory explanations: such phenomena are widely observed and therefore should be expected. For example, [45] review problems of expertise within the general machine learning community and [46] discusses problems of bias within bioinformatics. The problems associated with confirmation biases amongst scientists have been shown experimentally by Mynatt et al. [47] working with student scientists.

Bias and lack of blinding has been shown to be highly influential in other sciences such as emergency medicine where Bebarta et al. [48] report that the odds ratio of finding an effect was 3.2 (95% CI 1.3 to 7.7) when blinding was not or was employed. In other words when the treatment groups are known to the researcher they are three times more likely to report an effect. For computational studies where placebos are not applicable, blind analysis seems the obvious and compelling way forward.

The next question is how should the research community respond? There are at least three possibilities.

- First, we are working with increasingly sophisticated techniques and this is extends to the data pre-processing. Better communication and documentation is needed of the *details* of how classifier techniques have been used. The need for this has long been a problem in software engineering [49]. Improving communication and documentation will help protect against the other things which researcher groups are doing, i.e. the unwritten setup is just as important as that which is documented. A useful motivating example is the journal *Biostatistics* that has an associate editor for reproducibility who can assign grades of merit to conditionally accepted papers (D: data are available, C: code is available, and R: the code can be run produces the same results without much effort). In short we need to agree and use adequate reporting protocols.
- Joint comparative empirical studies of defect prediction between research centres such that researcher groups do not have to be expert in all classifier learning techniques.
- Last and very importantly we need to take blinding seriously. We need to remind ourselves that scientists are subject to the same extra-scientific pressures and sources of bias as others. Thus it is hard to think of any reason why blind analysis (that is analysing the results without knowing the treatments) should not be mandatory for all serious computational research.

It is not our intention to be unduly negative, however, the evidence is compelling and we cannot ignore the fact that the main determinant of a defect prediction experiment result is which researcher group undertakes the work. This cannot be ignored if the community are

to make meaningful progress on what, after all, is a very important problem of predicting software defects.

We briefly review potential threats to validity. ANOVA models generally make three assumptions [50] which are (i) normality, (ii) equality of variances, and (iii) random, independent sampling. We consider each of these issues in turn. A more detailed analysis, including a robust alternative to our statistical procedure is provided in the Supplementary Materials.

We have shown the response variable MCC is reasonably close to a normal distribution other than the extreme tails (see Fig. 4) and as discussed in the Supplementary Materials the standard corrective procedures are not helpful in this particular setting. Non-normality elevates the risk of a Type I error (i.e., wrongly rejecting the null hypothesis) and reduces power, however, as Harwell et al. ([51] - Table 7) suggest skewness may be problematic than kurtosis (which is what we observe for MCC in that the distribution is approximately symmetrical) but that the general impact is 'negligible' on both Type I errors and power.

The robust Levene test of distribution of variance is also satisfactory, furthermore, as Faraway comments "most tests and CI's [Confidence Intervals] are relatively insensitive to non-constant variance" [40].

The choice of studies for the meta-analysis is more complex, however, we have endeavoured to be exhaustive and have based the analysis on a previous comprehensive systematic literature review [2]. Thus all published studies that provide sufficient data for our meta-analysis and sufficient contextual information to be meaningfully compared are included. Consequently, we do not believe our analysis is biased in this regard. Moreover, the inclusion of interaction terms within the meta-analysis means that we can explicitly capture and account for dependencies between factors.

A second threat arise from the complexity of the 4-way model which means that from a factorial point of view many of the cells are empty since if we count the levels this yields ($23 \times 24 \times 7 \times 8 = 30912$). For this reason we looked at "dense" subsets of our data. We performed the analyses for both ECLIPSE and NASA datasets alone and by restricting it to the most popular class of Metric thereby reduced the model to two-way. Detailed results are given in the Supplementary Materials however to summarise Research Group remains dominant (accounting for between 5 and 60 times more variance than the choice of Classifier technique).

A third source of threat are our choices of category or class when we group specific values for each factor e.g. for data sets we ignore version. We do not see our process as being unduly subjective and are making the raw data available[3] to other researchers so that our findings can be corroborated, extended or challenged. In any case the levels of significance are so strong and the effects so large that we do not believe the basic findings

are sensitive to changes in the classification scheme. In addition it is a random-effects analysis so we focus on which factors matter rather than specific inter-level differences where choice of class might be more influential. Given, we have chosen fairly coarse-grained schemes our expectation is that effects would be magnified if more detailed classifications were employed.

A fourth threat is that concerns have been expressed about the quality of some of the data sets used, e.g., NASA and Eclipse [12], [52] and this might jeopardise the results. However, choice of data set would therefore be a proxy for quality and we would see its importance as a moderator increase. This may well be the case, however, Research Group remains the dominant factor. To further explore this question we conduct further studies based on NASA, Eclipse and Mozilla data sets alone and find no change in the main finding that Researcher Group is the dominant factor (see the Supplementary Materials).

Thus we conclude from the meta-analysis that there is strong evidence that current experiments in software defect prediction are extremely limited in their effectiveness and that we must take into account the dominant factor which is researcher group. Until we move on from the situation where it doesn't matter what you do, merely who does it, progress will be restricted since our research is not reproducible [53].

Finally, although our focus has been on dichotomous defect classifiers we anticipate that these kind of problems will be manifest in other areas where prediction systems are being researched. Given problems of bias have been reported across many branches of science it is difficult to imagine that other areas of software engineering will be immune. Research to confirm, or otherwise, this conjecture is urgently required.

ACKNOWLEDGMENTS

REFERENCES

[1] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Expert Systems with Applications*, vol. 36, no. 4, pp. 7346–7354, 2009.

[2] 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*, vol. 38, no. 6, pp. 1276 –1304, nov.-dec. 2012.

[3] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," *IEEE Transactions on Software Engineering*, vol. 34, no. 4, pp. 485–496, 2008.

[4] I. Myrtveit, E. Stensrud, and M. Shepperd, "Reliability and validity in comparative studies of software prediction models," *IEEE Transactions on Software Engineering*, vol. 31, no. 5, pp. 380–391, 2005.

[5] T. Raeder, T. Hoens, and N. Chawla, "Consequences of variability in classifier performance estimates," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, Dec 2010, pp. 421–430.

---

[3] https://codefeedback.cs.herts.ac.uk/mlbias.

[6] Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu, "A general software defect-proneness prediction framework," *IEEE Transactions on Software Engineering*, vol. 11, no. 3, pp. 356–370, May 2011.

[7] D. Ince, L. Hatton, and J. Graham-Cumming, "The case for open computer programs," *Nature*, vol. 482, pp. 485–488, 2012.

[8] P. Baldi, S. Brunak, Y. Chauvin, C. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: an overview," *Bioinformatics*, vol. 16, no. 5, pp. 412–424, 2000.

[9] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," *International Joint Conference on Artificial Intelligence*, 1995.

[10] J. D. Rodríguez, A. Pérez, and J. A. Lozano, "Sensitivity analysis of k-fold cross validation in prediction error estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 3, pp. 569–75, 2010.

[11] T. Menzies, B. Caglayan, Z. He, E. Kocaguneli, J. Krall, F. Peters, and B. Turhan. (2012, June) The PROMISE repository of empirical software engineering data. [Online]. Available: http://promisedata.googlecode.com

[12] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson, "The misuse of the NASA metrics data program data sets for automated software defect prediction," in *EASE 2011*. Durham, UK: IET, 2011.

[13] M. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: Some comments on the nasa software defect datasets," *Software Engineering, IEEE Transactions on*, vol. 39, no. 9, pp. 1208–1215, Sept 2013.

[14] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project defect prediction: a large scale experiment on data vs. domain vs. process," 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*, ser. ESEC/FSE '09. New York, NY, USA: ACM, 2009, pp. 91–100.

[15] E. Arisholm, L. C. Briand, and E. B. Johannessen, "A systematic and comprehensive investigation of methods to build and evaluate fault prediction models," Simula, Tech. Rep. TR-2008-06, 2008.

[16] D. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[17] Y. Jiang, B. Cukic, and Y. Ma, "Techniques for evaluating fault prediction models," *Empirical Software Engineering*, vol. 13, pp. 561–595, 2008.

[18] J. Hernández-Orallo, P. Flach, and C. Ferri, "A unified view of performance metrics: Translating threshold choice into expected classification loss," *Journal of Machine Learning Research*, vol. 13, pp. 2813–2869, 2012.

[19] G. Forman and M. Scholz, "Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement," *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 1, 2010.

[20] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering, version 2.3," Keele University, UK, Tech. Rep. EBSE Technical Report EBSE-2007-01., 2007.

[21] M. Blettner, W. Sauerbrei, B. Schlehofer, T. Scheuchenpflug, and C. Friedenreich, "Traditional reviews, meta-analyses and pooled analyses in epidemiology," *Intl. J. of Epidemiology*, vol. 28, no. 1, pp. 1–9, 1999.

[22] G. V. Glass, "Primary, secondary, and meta-analysis of research," *Educational Researcher*, vol. 5, no. 10, pp. 3–8, 1976.

[23] L. Pickard, B. Kitchenham, and P. Jones, "Combining empirical results in software engineering," *Information & Software Technology*, vol. 40, no. 14, pp. 811–821, 1998.

[24] J. Miller, "Applying meta-analytical procedures to software engineering experiments," *J. of Systems & Software*, vol. 54, no. 1, pp. 29–39, 2000.

[25] D. Cruzes and T. Dybå, "Research synthesis in software engineering: A tertiary study," *Information and Software Technology*, vol. 53, no. 5, pp. 440–455, 2011.

[26] M. Egger, M. Schneider, and G. Davey-Smith, "Meta-analysis spurious precision? Meta-analysis of observational studies," *BMJ*, vol. 316, p. 140, 1998.

[27] S. Shapiro, "Meta analysis/shmeta analysis," *American J. of Epidemiology*, vol. 140, pp. 771–778, 1994.

[28] G. Glass, B. McGaw, and M. Smith, *Meta-analysis in social research*. Beverly Hills, CA: Sage Publications, 1981.

[29] K. Petersen and C. Wohlin, "Context in industrial software engineering research," in *3rd IEEE Intl. Symp on Empirical Software Engineering & Measurement (ESEM)*, 2009.

[30] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[31] M. Warrens, "On association coefficients for 2×2 tables and properties that do not depend on the marginal distributions," *Psychometrika*, vol. 73, no. 4, pp. 777–789, 2008.

[32] E. Davenport and N. El-Sanhurry, "Phi/Phimax: Review and synthesis," *Educational and Psychological Measurement*, vol. 51, pp. 821–828, 1991.

[33] D. Bowes, T. Hall, and D. Gray, "DConfusion: A technique to allow cross study performance evaluation of fault prediction studies." *Automated Software Engineering*, 2013.

[34] S. Chidamber and C. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476–493, 2011.

[35] S. Siegel and N. Castellan, "Nonparametric statistics for the behavioral sciences (Mcgraw-hill, New York)," 1988.

[36] 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*, vol. 2, dec. 2012, pp. 308 –313.

[37] M. Shepperd and G. Kadoda, "Comparing software prediction techniques using simulation," *IEEE Trans. on Softw. Eng.*, vol. 27, no. 11, pp. 987–998, Nov 2001.

[38] K. Krishnamoorthy, F. Lu, and T. Matthew, "A parametric bootstrap approach for ANOVA with unequal variances: Fixed and random models," *Computational Statistics and Data Analysis*, vol. 51, no. 12, pp. 5731–5742, 2007.

[39] T.-S. Lim and W.-Y. Loh, "A comparison of tests of equality of variances," *Computational Statistics & Data Analysis*, vol. 22, no. 3, pp. 287–301, 1996.

[40] J. J. Faraway, "Practical regression and anova using r." 2002.

[41] T. Levine and C. Hullett, "Eta squared, partial eta squared, and misreporting of effect size in communication research," *Human Communication Research*, vol. 28, no. 4, pp. 612–625, 2002.

[42] C. Pierce, R. Block, and H. Aguinis, "Cautionary note on reporting eta-squared values from multifactor ANOVA designs," *Educational and Psychological Measurement*, vol. 64, no. 6, pp. 916–924, 2004.

[43] B. Kitchenham and E. Mendes, "Why comparative effort prediction studies may be invalid," in *Proceedings of the 5th International Conference on Predictor Models in Software Engineering*. ACM, 2009, p. 4.

[44] M. Shepperd, "It doesn't matter what you do but does matter who does it!" UCL, London, 2011. [Online]. Available: crest.cs.ucl.ac.uk/15/slides/MartinShepperdCOW15.pptx

[45] D. Michie, D. Spiegelhalter, and C. Taylor, Eds., *Machine learning, neural and statistical classification*, ser. Ellis Horwood Series in Artificial Intelligence. Chichester, Sussex, UK: Ellis Horwood, 1994.

[46] A.-L. Boulesteix, "Over-optimism in bioinformatics research," *Bioinformatics*, vol. 26, no. 3, pp. 437–439, 2010.

[47] C. Mynatt, M. Doherty, and R. Tweney, "Confirmation bias in a simulated research environment: An experimental study of scientific inference," *Quarterly Journal of Experimental Psychology*, vol. 29, no. 1, pp. 85–95, 1977.

[48] V. Bebarta, D. Luyten, and K. Heard, "Emergency medicine animal research: does use of randomization and blinding affect the results?" *Academic Emergency Medicine*, vol. 10, no. 6, pp. 684–687, 2003.

[49] B. Kitchenham, "What's up with software metrics? - A preliminary mapping study," *J. Syst. Softw.*, vol. 83, no. 1, pp. 37–51, Jan. 2010.

[50] B. West, K. Welch, and A. Galecki, *Linear Mixed Models: A Practical Guide Using Statistical Software*, ser. Chapman Hall/CRC Press. Boca Raton, FL: Ellis Horwood, 2007.

[51] M. Harwell, E. Rubinstein, W. Hayes, and C. Olds, "Summarizing monte carlo results in methodological research: The one- and two-factor fixed effects anova cases," *Journal of Educational and Behavioral Statistics*, vol. 17, no. 4, pp. 315–339, 1992.

[52] C. Bird, A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov, and P. 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*, ser. ESEC/FSE '09. New York, NY, USA: ACM, 2009, pp. 121–130.

[53] C. Drummond, "Replicability is not reproducibility: Nor is it good science," June 14-18, 2009 2009.

SECONDARY LITERATURE

[S1] G. Calikli, A. Tosun, A. Bener, and M. Celik, "The effect of granularity level on software defect prediction," in *Computer and Information Sciences, 2009. ISCIS 2009. 24th International Symposium on*, sept. 2009, pp. 531 –536.

[S2] B. Caglayan, A. Bener, and S. Koch, "Merits of using repository metrics in defect prediction for open source projects," in *Emerging Trends in Free/Libre/Open Source Software Research and Development, 2009. FLOSS '09. ICSE Workshop on*, 2009, pp. 31–36.

[S3] O. Kutlubay, B. Turhan, and A. Bener, "A two-step model for defect density estimation," in *Software Engineering and Advanced Applications, 2007. 33rd EUROMICRO Conference on*, aug. 2007, pp. 322 –332.

[S4] 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*, oct. 2007, pp. 231 –237.

[S5] A. Tosun and A. Bener, "Reducing false alarms in software defect prediction by decision threshold optimization," in *Empirical Software Engineering and Measurement, 2009. ESEM 2009. 3rd International Symposium on*, 2009, pp. 477–480.

[S6] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *Software Engineering, IEEE Transactions on*, vol. 33, no. 1, pp. 2 –13, jan. 2007.

[S7] A. Kaur, P. S. Sandhu, and A. S. Bra, "Early software fault prediction using real time defect data," in *Machine Vision, 2009. ICMV '09. Second International Conference on*, 2009, pp. 242–245.

[S8] A. Kaur and R. Malhotra, "Application of random forest in predicting fault-prone classes," in *Advanced Computer Theory and Engineering, 2008. ICACTE '08. International Conference on*, 2008, pp. 37–43.

[S9] Y. Singh, A. Kaur, and R. Malhotra, "Predicting software fault proneness model using neural network," *Product-Focused Software Process Improvement*, vol. 5089, pp. 204–214, 2008.

[S10] G. Pai and J. Dugan, "Empirical analysis of software fault content and fault proneness using bayesian methods," *Software Engineering, IEEE Transactions on*, vol. 33, no. 10, pp. 675 –686, oct. 2007.

[S11] C. Bird, N. Nagappan, H. Gall, B. Murphy, and P. Devanbu, "Putting it all together: Using socio-technical networks to predict failures," in *20th International Symposium on Software Reliability Engineering*. IEEE, 2009, pp. 109–119.

[S12] N. Nagappan, A. Zeller, T. Zimmermann, K. Herzig, and B. Murphy, "Change bursts as defect predictors," in *Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium on*, 2010, pp. 309–318.

[S13] T. Zimmermann, R. Premraj, and A. Zeller, "Predicting defects for eclipse," in *Predictor Models in Software Engineering, 2007. PROMISE'07: ICSE Workshops 2007. International Workshop on*, may 2007, p. 9.

[S14] A. Schröter, T. Zimmermann, and A. Zeller, "Predicting component failures at design time," in *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*. ACM, 2006, pp. 18–27.

[S15] A. Koru and H. Liu, "Building effective defect-prediction models in practice," *Software, IEEE*, vol. 22, no. 6, pp. 23 – 29, nov.-dec. 2005.

[S16] G. Denaro and M. Pezzè, "An empirical evaluation of fault-proneness models," in *Proceedings of the 24th International Conference on Software Engineering*, ser. ICSE '02. New York, NY, USA: ACM, 2002, pp. 241–251.

[S17] A. B. de Carvalho, A. Pozo, S. Vergilio, and A. Lenz, "Predicting fault proneness of classes trough a multiobjective particle swarm optimization algorithm," in *Tools with Artificial Intelligence, 2008. ICTAI '08. 20th IEEE International Conference on*, vol. 2, 2008, pp. 387–394.

[S18] A. B. de Carvalho, A. Pozo, and S. R. Vergilio, "A symbolic fault-prediction model based on multiobjective particle swarm optimization," *Journal of Systems and Software*, vol. 83, no. 5, pp. 868–882, 2010.

[S19] A. Nugroho, M. R. V. Chaudron, and E. Arisholm, "Assessing uml design metrics for predicting fault-prone classes in a java system," in *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*, 2010, pp. 21–30.

[S20] L. Briand, W. Melo, and J. Wust, "Assessing the applicability of fault-proneness models across object-oriented software projects," *Software Engineering, IEEE Transactions on*, vol. 28, no. 7, pp. 706 – 720, jul 2002.

[S21] E. Arisholm, L. C. Briand, and M. Fuglerud, "Data mining techniques for building fault-proneness models in telecom java software," in *Software Reliability, 2007. ISSRE '07. The 18th IEEE International Symposium on*, nov. 2007, pp. 215 –224.

[S22] E. Arisholm, L. C. Briand, and E. B. Johannessen, "A systematic and comprehensive investigation of methods to build and evaluate fault prediction models," *Journal of Systems and Software*, vol. 83, no. 1, pp. 2–17, 2010.

[S23] C. Cruz and A. Erika, "Exploratory study of a uml metric for fault prediction," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*. ACM, 2010, pp. 361–364.

[S24] L. Yi, T. M. Khoshgoftaar, and N. Seliya, "Evolutionary optimization of software quality modeling with multiple repositories," *Software Engineering, IEEE Transactions on*, vol. 36, no. 6, pp. 852–864, 2010.

[S25] N. Seliya, T. Khoshgoftaar, and S. Zhong, "Analyzing software quality with limited fault-proneness defect data," in *High-Assurance Systems Engineering, 2005. HASE 2005. Ninth IEEE International Symposium on*, oct. 2005, pp. 89 –98.

[S26] T. Khoshgoftaar and N. Seliya, "Comparative assessment of software quality classification techniques: An empirical case study," *Empirical Software Engineering*, vol. 9, no. 3, pp. 229–257, 2004.

[S27] T. Khoshgoftaar, X. Yuan, E. Allen, W. Jones, and J. Hudepohl, "Uncertain classification of fault-prone software modules," *Empirical Software Engineering*, vol. 7, no. 4, pp. 297–318, 2002.

[S28] T. Mende and R. Koschke, "Effort-aware defect prediction models," in *Software Maintenance and Reengineering (CSMR), 2010 14th European Conference on*, 2010, pp. 107–116.

[S29] C. Catal, B. Diri, and B. Ozumut, "An artificial immune system approach for fault prediction in object-oriented software," in *Dependability of Computer Systems, 2007. DepCoS-RELCOMEX '07. 2nd International Conference on*, june 2007, pp. 238 –245.

[S30] 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*, 2008, pp. 181–190.

[S31] T. Gyimothy, R. Ferenc, and I. Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction," *Software Engineering, IEEE Transactions on*, vol. 31, no. 10, pp. 897 – 910, oct. 2005.

[S32] R. Vivanco, Y. Kamei, A. Monden, K. Matsumoto, and D. Jin, "Using search-based metric selection and oversampling to predict fault prone modules," in *Electrical and Computer Engineering (CCECE), 2010 23rd Canadian Conference on*, 2010, pp. 1–6.

[S33] Y. Ma, L. Guo, and B. Cukic, "A statistical framework for the prediction of fault-proneness," *Advances in machine learning application in software engineering*, pp. 237–265, 2006.

[S34] L. Guo, Y. Ma, B. Cukic, and H. Singh, "Robust prediction of fault-proneness by random forests," in *Software Reliability Engineering, 2004. ISSRE 2004. 15th International Symposium on*, nov. 2004, pp. 417 – 428.

[S35] Y. Jiang, B. Cukic, and Y. Ma, "Techniques for evaluating fault prediction models," *Empirical Software Engineering*, vol. 13, no. 5, pp. 561–595, 2008.

[S36] O. Mizuno, S. Ikami, S. Nakaichi, and T. Kikuno, "Spam filter based approach for finding fault-prone software modules," in *Mining Software Repositories, 2007. ICSE Workshops MSR '07. Fourth International Workshop on*, may 2007, p. 4.

[S37] O. Mizuno and T. Kikuno, "Training on errors experiment to detect fault-prone software modules by spam filter," in *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, ser. ESEC-FSE '07. New York, NY, USA: ACM, 2007, pp. 405–414.

[S38] Y. Zhou and H. Leung, "Empirical analysis of object-oriented design metrics for predicting high and low severity faults," *Software Engineering, IEEE Transactions on*, vol. 32, no. 10, pp. 771 –789, oct. 2006.

[S39] O. Vandecruys, D. Martens, B. Baesens, C. Mues, M. De Backer, and R. Haesen, "Mining software repositories for comprehens-

ible software fault prediction models," *Journal of Systems and Software*, vol. 81, no. 5, pp. 823–839, 2008.

[S40] Z. Hongyu, "An investigation of the relationships between lines of code and defects," in *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on*, 2009, pp. 274–283.

[S41] S. Kanmani, V. Uthariaraj, V. Sankaranarayanan, and P. Thambidurai, "Object-oriented software fault prediction using neural networks," *Information and Software Technology*, vol. 49, no. 5, pp. 483–492, 2007.

[S42] S. Shivaji, E. J. Whitehead, R. Akella, and K. Sunghun, "Reducing features to improve bug prediction," in *Automated Software Engineering, 2009. ASE '09. 24th IEEE/ACM International Conference on*, 2009, pp. 600–604.