Balancing Privacy and Utility in Cross-Company Defect Prediction

Fayola Peters, Tim Menzies, *Member*, *IEEE*, Liang Gong, and Hongyu Zhang, *Member*, *IEEE*

Abstract—*Background*: Cross-company defect prediction (CCDP) is a field of study where an organization lacking enough local data can use data from other organizations for building defect predictors. To support CCDP, data must be shared. Such shared data must be *privatized*, but that privatization could severely damage the utility of the data. *Aim:* To enable effective defect prediction from shared data while preserving privacy. *Method:* We explore privatization algorithms that maintain class boundaries in a dataset. CLIFF is an instance pruner that deletes irrelevant examples. MORPH is a data mutator that moves the data a random distance, taking care not to cross class boundaries. CLIFF+MORPH are tested in a CCDP study among 10 defect datasets from the PROMISE data repository. *Results:* We find: 1) The CLIFFed+MORPHed algorithms provide *more* privacy than the state-of-the-art privacy algorithms; 2) in terms of utility measured by defect prediction, we find that CLIFF+MORPH performs significantly better. *Conclusions:* For the OO defect data studied here, data can be privatized and shared without a significant degradation in utility. To the best of our knowledge, this is the first published result where privatization does not compromise defect prediction.

Index Terms—Privacy, classification, defect prediction

1 Introduction

Within-company defect prediction (WCDP) is the means by which organizations predict the number of defects in their software. *Cross-company defect prediction* (CCDP) looks at the feasibility of learning defect predictors using data from other companies. Recent studies show that defect and effort predictors built from *cross-company data* can be just as effective as predictors learned using *within-company data* [1], [2], [3] (caveat: the cross-company data must be carefully filtered before being applied locally). This is potentially a very important result that implies the existence of general principles of software engineering (SE) (such generalities would lead us to general models of SE).

However, before we conduct widespread CCDP experiments, we must address the privacy concerns of data owners. Extracting data from organizations is often difficult due to the business sensitivity associated with the data. Because of this sensitivity, data owners who want to share limited amounts of useful data (say, to advance scientific research leading to improved software) need to do so without breaching any data privacy laws or company privacy policies. Unless we can address these concerns, continued progress in this promising area of CCDP will be stalled.

For these reasons, many researchers doubt the practicality of data sharing for the purposes of research. In a personal communication, Barry Boehm reports he can release none of the data that his COCOMO team collected after 1981. Similarly, at a recent keynote address at ESEM '11, Elaine Weyuker doubted that she will ever be able to release the AT&T data she used to build defect predictors [4].

For companies with common concerns (e.g., subcontractors for a primary company), the benefits of sharing data can include improved software quality and reduced SE costs. Ideally, these organizations should be able to share data without revealing too much. Such organizations need to

1. prevent the disclosure of specific sensitive metric values from their released data, and
2. ensure that the privatized data remains useful for research purposes such as CCDP.

To date, the research community has *not* achieved these goals. As shown below, standard methods such as k-anonymity do not protect against any background knowledge of an *attacker* (see Table 1 for a definition of this and other terms used in this work) and so may still reveal the sensitive attribute of a record. Moreover, two recent reports concluded that the *more* we privatize data, the *less* useful it becomes for certain utilities of certain tasks, for example, *classification*. Grechanik et al. [5] and Brickell and Shmatikov [6] reported that the application of standard privacy methods such as k-anonymity, l-diversity, and t-closeness damages inference power as privacy increases.

This paper proposes two privatization algorithms. The CLIFF instance pruner finds attribute subranges in the data that are more present in one class versus any other classes. These *power subranges* are those that drive instances furthest from the *class boundaries*. If an instance lacks these power subranges, then their classification is more ambiguous. CLIFF deletes the instances with the fewest power subranges.

After CLIFF, the MORPH instance mutator perturbs all instance values by a random amount. This amount is

• F. Peters and T. Menzies are with the Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV 26506-610. E-mail: fayolapeters@gmail.com, tim@menzies.us.
• L. Gong and H. Zhang are with the School of Software, Tsinghua University, Beijing 100084, China.
E-mail: gongliang10@mails.tsinghua.edu.cn, hongyu@tsinghua.edu.cn.

Manuscript received 31 Mar. 2012; revised 4 Oct. 2012; accepted 27 Dec. 2012; published online 23 Jan. 2013.
Recommended for acceptance by J. Cleland-Huang.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-2012-03-0078.
Digital Object Identifier no. 10.1109/TSE.2013.6.

TABLE 1
Definitions of Terms Used in This Paper

Terms	Definitions
CCDP	Cross Company Defect Prediction.
WCDP	Within Company Defect Prediction.
Classification	• For a data-set containing independent attributes and one dependent attribute (class). • We find a model for the dependent attribute as a function of the independent attributes. • The goal is to assign the right class value to an unseen instance. • Defect prediction is an example of a classification task.
class boundary	the border between neighboring regions of different classes.
EFB	Equal Frequency Binning. EFB divides the range of possible values into n bins or sub-ranges, each of which holds the same number of attribute values. If duplicate values are placed into different bins, boundaries of every pair of neighboring bins are adjusted so that duplicate values belong to one bin only [9].
CLIFF	An instance selection method.
power sub-range	power of each attribute sub-range; i.e., how much more frequently that sub-range appears in one class more than any other.
MORPH	Data transformation via perturbation to create synthetic data
t_i	A tuple of attribute values representing an individual record in a data-set. Also referred to as a $target$ in this work.
T	Original data-set containing targets.
T'	Privatized data-set.
T^*	Represents either T or T'
A	Set of attributes.
a_i	Attribute.
$t[a_i]$	The value of attribute a_i for t.
ID	Identifier - anything that specifically identifies an individual or thing such as a social security number
QIDs	Quasi-Identifiers are attributes that can be used to re-identify an individual record in a data-set. These attributes can also be sensitive.
s_i	Sensitive attribute value.
S	Set of possible sensitive attribute values that we do not want an attacker to associate with a target, t in a data-set.
q_i	A query is made up of attribute value pair(s). Example q_3 is "$a_1 = t[a_1]$ and $a_2 = t[a_2]$". It represents information an attacker has about a specific target in a data-set.
$\|q_i\|$	Query sizes in this work are 1, 2, and 4. Example, $\|q_3\| = 2$.
Q	Set of Queries generated randomly from T. In this work $\|Q\| \leq 1000$.
G^*	Group of t's from either the original or privatized data-set that matches a query, q_i. G_i = t's from the original data-set and G'_i = t's from the privatized data-set.
$s_{max}(G_i)$	Most common sensitive attribute value in G_i.
$s_{max}(G'_i)$	Most common sensitive attribute value in G'_i.
$Breach(S, G^*_i)$	Privacy breach $$Breach(S, G^*_i) = \begin{cases} 1, & if \quad s_{max}(G_i) = s_{max}(G'_i), \\ 0, & otherwise. \end{cases}$$
$IPR(T^*)$	Increase Privacy Ratio $$IPR(T^*) = 1 - \frac{1}{\|Q\|}\sum_{i=0}^{\|Q\|} Breach(S, G^*_i).$$

Terms that are related to each other are grouped together.

selected to create a new instance which is different from the original instance and does not cross class boundaries.

Potentially, CLIFF+MORPH can increase privacy in three ways. First, CLIFF preserves the privacy of the individuals it deletes (since these data are no longer available). Second, MORPH increases the privacy of all mutated individuals since their original data is now distorted. Finally, to ensure that MORPHed instances are different from instances in the original dataset, if a MORPHed instance matches an instance from the original data, it is either MORPHed again until it no longer matches the original or it is removed.

To assess the privacy and utility of CLIFF+MORPH, we explore three research questions.

- RQ1: Does CLIFF+MORPH provide better balance between privacy and utility than other state-of-the-art privacy algorithms?
- RQ2: Do different classifiers affect the experimental results?
- RQ3: Does CLIFF+MORPH perform better than MORPH?

1.1 Results and Contributions

The experiments of this paper show that after applying CLIFF+MORPH, both the efficacy of classification and privacy are increased. Also, combining CLIFF and MORPH improves on prior results. Previously, Peters and Menzies [7] used MORPH and found that, sometimes, the privatized data exhibited worse performance than the original data. In this study, we combine CLIFF and MORPH and show that there are no significant reductions in the classification performance in any of the datasets we study. Note that these are *CCDP results* where data from many outside companies were combined to learn defect predictors for a local company.

Further, it is well known that pruning before classifying usually saves time [8]. For instance, *tomcat*, the largest dataset used in the study, is more than 163 times faster when CLIFF prunes away 90 percent of the data prior to MORPHing. That is, using this combination of CLIFF and MORPH, we achieve *more* privacy for effective CCDP in *less* time.

These results are novel and promising. To the best of our knowledge, this is the first report of a privacy algorithm that increases privacy while preserving inference power. Hence, we believe CLIFF+MORPH is a better option for preserving privacy in a scenario where data is shared for the purpose of CCDP.

1.2 Relation to Prior Publications

This paper extends a prior publication [7] in five ways, First, as said above, this paper shows that combining CLIFF and MORPH is better than just running MORPH.

Second, in addition to naive Bayes (NB), we experimented with two additional classification techniques (neural networks (NNs)) and support vector machines (SVMs)).

Third, in addition to data swapping, we include experiments with *k*-anonymity.

Fourth, based on reviewer feedback, we have improved our increased privacy ratio (IPR) measure such that it now uses $100\% - \%$ of correct guesses or breaches. (See Table 1 for more details.)

Finally, this paper has more extensive descriptions of our motivation, algorithms, and related work.

1.3 Organization

The remainder of this paper is structured as follows: Table 1 lists the key terms of this paper. We present background on WCDP, CCDP, utility measures, and privacy in Section 2. Next, we describe the problems with privacy in Section 3, followed by our solution and privacy algorithms in Section 4. In Section 5, the means of assessing privacy is explained and in Section 6 experimental procedure and results are presented. Finally, threats to validity, related work, the conclusion, and future work are discussed in Sections 7, 8, and 9, respectively.

2 Background

2.1 Within Company Defect Prediction

Defect prediction allows software companies to take advantage of early defect detection. It allows for the focusing the QA budgets on where it might be most cost effective [10]. This is an important task as, during development, developers have to *skew* their quality assurance activities toward artifacts that they believe require the most effort due to limited project resources.

Models made for defect prediction are built with within-company (local) datasets using common classification techniques [11], [12]. The datasets are comprised of independent variables such as the code metrics used in this work and one dependent variable or prediction target with values (labels) to indicate whether or not defects are present. A prediction model created from a defect dataset can then take a new unlabeled instance and label it as defective or not defective.

2.2 Cross-Company Defect Prediction

When data can be shared between organizations, static code defect predictors from one organization can generalize to another. For example, defect predictors developed at NASA [11] have also been used in software development companies outside the US (in Turkey). When the inspection teams focused on the modules that trigger the defect predictors, they found up to 70 percent of the defects using just 40 percent of their QA effort (measured in staff hours) [13].

Such CCDP is useful since, as Zimmermann et al. [14] observed, defect prediction via local data is not always available to many software companies as

- the companies may be too small and
- the product might be in its first release and so there is no past data.

Kitchenham et al. [15] also saw problems with relying on within-company datasets. They noted that the time required to collect enough data on past projects from a single company may be prohibitive. Additionally, they stated that collecting within-company data may take so long that technologies used by the company would have changed and therefore older projects may no longer represent current practices.

Initial experiments with CCDP were either very negative [14] or inconclusive [15]. Recently, we have had more success using better selection tools for training data [2], [3], but this success was only possible if classifying technique had unrestricted access to all the data. As discussed below, this is a problem.

2.3 Utility Measures

The defect data used in this work are mainly utilized via classification. The goal is to predict if an unlabeled instance is *defective* or *not defective*. However, our proposed privacy algorithms can work for other classification tasks as well. These tasks can be applied to data that is extracted from software. For example, the proposed approach can be used to classify software applications to assign them to different semantic categories like *financial* and *games*, or as in our case *defective* or *not defective*.

Since our research focuses on defect data, we elaborate on defect prediction in the following sections.

2.4 Privacy and CCDP

Data sharing across companies exposes the data provider to unwanted risks. Some of these concerns reflect the low quality of our current anonymization technologies. For example, the state of Massachusetts once released some health care data anonymized according to HIPPA regulations [16]. When this "anonymized" data was joined to other data (Massachusetts' list of registered voters) it was possible to identify which health care data corresponded to specific individuals (e.g., former Massachusetts governor William Weld [17]).

We say that *reidentification* occurs when an attacker with external information such as a voters' list can identify an individual from a privatized dataset. The datasets used in this study are aggregated at the project level and do not contain personnel or company information. Hence, reidentification of individuals is not explored further in this study.

On the other hand, *sensitive attribute disclosure* is of great concern with the data used in this study. This is where an individual in a dataset can be associated with a sensitive attribute value; for example, software development time. Such sensitive attribute disclosure can prove problematic. Some of the metrics contained in defect data can be considered as *sensitive* to the data owners. These can include lines of code (loc) or cyclomatic complexity (max-cc or avg-cc).[1] If these software measures are joined to development time, privacy policy may be breached by revealing (say) slow development times.

3 Problems with Privacy

Many researchers comment on how privatization algorithms can distort data. For example, consider privatization via generalization and suppression. Generalization can be done by replacing exact numeric values with intervals that cover a subrange of values, for example, 17 might become 15..20, or by replacing symbols with more general terms, for example, "date of birth" becomes "month of birth." Suppression can be done by replacing exact values with symbols such as a *star* or a phrase like "don't know" [21]. According to Fung et al. [19], generalization and suppression hide potentially important details in the QIDs that can confuse classification. Worse, these transforms may not guarantee privacy. For example, consider privacy-via-perturbation. Suppose an attacker has access to multiple independent samples from the same distribution from which the original data was drawn. In that case, a principal component analysis could reconstruct the transform from the original to privatized data [22]. Here, the attacker's goal is to estimate the matrix (M_T) used to transform the original

1. In future work, we will explore multiple sensitive attributes. For this paper, we scope this work to data with a single sensitive attribute.

data to its privatized version. M_T is then used to undo the data perturbation applied to the original data. According to Giannella et al. [22], $M_T = WD_0Z'$, where W is the eigenvector matrix of the covariance matrix of the privatized data. Z' is the transform of the eigenvector matrix of the covariance matrix of the independent samples. Finally, D_o is an identity matrix.

Widely used privatization approaches include k-anonymity, l-diversity, t-closeness, and ϵ-differential. k-anonymity [17] makes each record in the table indistinguishable from $k-1$ other records by suppression or generalization [17], [23], [24]. The limitations of k-anonymity, as listed by Brickell and Shmatikov [6], are many-fold. They state that k-anonymity does not hide whether a given individual is in the database. Also, in theory, k-anonymity hides uniqueness (and hence identity) in a dataset, thus reducing the certainty that an attacker has uncovered sensitive information. However, in practice, k-anonymity does not ensure privacy if the attacker has background knowledge of the domain (see Fig. 1).

Machanavajjhala et al. [25] proposed l-diversity. The aim of l-diversity is to address the limitations of k-anonymity by requiring that for each QID group[2] there are at least l distinct values for each sensitive attribute value. In this way, an attacker is less likely to "guess" the sensitive attribute value of any member of a QID group.

Work by Li and Ruhe [26] later showed that l-diversity was vulnerable to *skewness* and *similarity* attacks, making it insufficient to prevent attribute disclosure. Hence, Li and Ruhe proposed t-closeness to address this problem. t-closeness focuses on keeping the distance between the distributions of a sensitive attribute in a QID group and that of the whole table no more than a threshold t apart. The intuition is that even if an attacker can locate the QID group of the target record, as long as the distribution of the sensitive attribute is similar to the distribution in the whole table, any knowledge gained by the attacker cannot be considered as a privacy breach because the information is already public. However, with t-closeness, information about the correlation between QIDs and sensitive attributes is limited [26] and so causes degradation of data utility.

According to Fung et al. [19], ϵ-differential privacy is based on the idea that the risk to the record owners privacy should not substantially increase as a result of participating in a statistical database. So, instead of comparing the prior probability and the posterior probability before and after accessing the published data, Dwork [27], [28] proposed to compare the risk with and without the record owners data in the published data.

Dwork [27], [28] defines ϵ-differential privacy as follows:

> We say databases D1 and D2 differ in at most one element if one is a proper subset of the other and the larger database contains just one additional row.

Differential privacy falls into the category of *output perturbation*, where it is achieved by adding noise to the outcome of a query. Work by Dinur and Nissim [29] also falls into this category and forms the basis of Dwork's work on differential privacy [27], [28].

2. A QID group is a set of instances whose QID values are the same because of generalization or suppression.

Consider the abbreviated *ant-1.3* data shown in the following table from the PROMISE data repository [18]. Descriptions of the *QID*s can be found in Table 5. We assume that we want to share this data.

ID	QIDs								
name	wmc	dit	noc	cbo	rfc	lcom	ca	ce	loc^*
taskdefs.ExecuteOn	11	4	2	14	42	29	2	12	395
DefaultLogger	14	1	1	8	32	49	4	4	257
taskdefs.TaskOut-putStream	3	2	0	1	9	0	0	1	58
taskdefs.Cvs	12	3	0	12	37	32	0	12	310
taskdefs.Copyfile	6	3	0	4	21	1	0	4	136
types.Enumerated-Attribute	5	1	5	12	11	8	11	1	59
NoBannerLogger	4	2	0	3	16	0	0	3	59

The ten attributes of this table divide into two categories: *Identifier* ID and *Quasi-Identifiers* QIDs. Note that the sensitive QID is indicated by a superscript "*", in this case it's lines of code (loc). An ID could be anything that specifically identifies an individual or thing such as a social security number, first and last names or a filename. Unlike an ID, QIDs are not specific to a particular individual or thing. However, they can be used to re-identify an individual record in a database.

The first step in privatizing this data-set is to de-identify the table i.e., remove the identity attribute "name" (but note that for the ease of explanation, we leave the ID for the example). One might think that removing the ID column should be enough to protect individual privacy, however, research has shown that this is not the case [6], [17], [19], [20]. In fact, using external public databases and/or personal background knowledge, an attacker can re-identify an individual record and associate that record with a sensitive attribute. For example, suppose the attacker has the following background knowledge.

$$rfc = 11 \text{ or } lcom = 0$$

On the data with deleted IDs, this attacker might use this knowledge to select the rows containing *type.EnumeratedAttribute*, *taskdefs.TaskOutputStream*, and *NoBannerLogger*, thereby learning with 100% certainty that there are 58 or 59 lines of code in the target file.

Even after applying a privacy algorithm, that background knowledge can still be used to violate privacy. Suppose this attacker studies the following *k=2*-anonymous version of the above data:

ID	QIDs								
name	wmc	dit	noc	cbo	rfc	lcom	ca	ce	loc^*
taskdefs. ExecuteOn	11-14	<5	≤ 5	8-14	32-42	29-49	*	*	395
taskdefs.Cvs	11-14	<5	≤ 5	8-14	32-42	29-49	*	*	310
Default Log-ger	11-14	<5	≤ 5	8-14	32-42	29-49	*	*	257
taskdefs. TaskOut-putStream	<7	<5	≤ 5	1-4	*	≤ 8	0	≤ 4	58
taskdefs. Copyfile	<7	<5	≤ 5	1-4	*	≤ 8	0	≤ 4	136
types. Enumerated Attribute	<7	<5	≤ 5	*	11-16	≤ 8	*	≤ 4	59
NoBanner Logger	<7	<5	≤ 5	*	11-16	≤ 8	*	≤ 4	59

"*" denotes that any value is possible

The background knowledge that $rfc = 11$ and $lcom = 0$ will result in 4 records being returned (the last four rows). With this result, because of the lack of diversity in the sensitive attribute of the result, an attacker will know with 75% certainty that the target has 58 or 59 lines of code.

Fig. 1. Effects of background knowledge on privacy.

Although ϵ-differential privacy assures record owners that they may submit their personal information to the database securely, it does not prevent membership disclosure and sensitive attribute disclosure studied in this work. This is shown in an example from Chin and Klinefelter [30] in a Facebook advertiser case study. Through reverse engineering, Chin and Klinefelter [30] inferred that Facebook uses differential privacy for its targeted advertising system.

To illustrate the problem of membership and sensitive attribute disclosure, the authors described Jane's curiosity about her neighbor John's HIV status when she learned that he was on the finishers list for the 2011 Asheville AIDS Walk and 5K Run. So, armed with John's age and zip code, she went to Facebook's targeted advertising area and found that there was exactly one male Facebook user age 36 from zip code 27514 who listed the "2011 Asheville AIDS Walk and 5K Run" as an interest. At this point, Jane placed a targeted advertisement offering free information to HIV-positive patients about a new antiretroviral treatment. If charged by Facebook for having her ad clicked, Jane can assume with some level of certainty that John is HIV positive.

In practice, the above issues with privacy algorithms are very real problems. Grechanik et al. [5] found that k-anonymity greatly degraded the test coverage of data-centric applications. Furthermore, Brickell and Shmatikov [6] reported experiments where achieving privacy using the above methods "requires almost complete destruction of the data mining capability." They concluded that depending on the privatization parameter, the privatized data provided no additional utility versus trivial privatization that privatizes data by simply removing all the sensitive attributes or all the other QIDs.

Worse, they also reported that simplistic trivial privatization provides better privacy results than supposedly better methods like l-diversity, t-closeness, and k-anonymity.

4 Our Approach

This section describes how we use CLIFF and MORPH to privatize datasets. As mentioned in Section 1, privatization with CLIFF+MORPH starts with removing a certain percentage of the original data. For our experiments, we remove 90, 80, and 60 percent of the original data with CLIFF (see Section 4.1). The remaining data is then MORPHed (see Section 4.2). The result is a privatized dataset with fewer instances, none of which could be found in the original dataset. When we perform our CCDP experiments, we combine multiple CLIFFed+MORPHed datasets to create a defect predictor which is used to predict defects in nonprivatized datasets. An example of how CLIFF and MORPH work together is provided in Section 4.3.

4.1 CLIFF

CLIFF is an instance pruner that assumes tables of training data can be divided into *classes*. For example, for a table of defect data containing code metrics, different rows might be labeled accordingly (defective or not defective).

CLIFF executes as follows:

- For each column of data, find the *power* of each attribute subrange, i.e., how much more frequently that subrange appears in one class more than any other.
- In prior work [31], at this point we select the subrange with the highest *power* and removed all instances without this subrange. From the remaining instances, those with subranges containing the second highest *power* are kept while the others are removed. This process continued until at least two instances were left or to the point before there were zero instances left. In this work, to control the amount of instances left by CLIFF, we find the product of the *powers* for each row, then
- Remove the less *powerful* rows.

The result is a reduced dataset with fewer rows. In theory, this reduced dataset is less susceptible to privacy breaches.

Algorithm 1. *Power* is based on BORE [32]. First, we assume that the target class is divided into one class as *first* and the other classes as *rest*. This makes it easy to find the attribute values which have a high probability of belonging to the current *first* class using Bayes theorem. The theorem uses evidence E and a prior probability $P(H)$ for hypothesis $H \in \{first, rest\}$ to calculate a likelihood (hereafter, *like*) of the evidence selecting for one class:

$$like(H|E) = P(E|H) \times P(H).$$

This calculation is then normalized to create probabilities:

$$P(first|E) = \frac{like(first|E)}{like(first|E) + like(rest|E)}. \tag{1}$$

Jalali et al. [32] found that (1) was a poor ranking heuristic for low frequency evidence. To alleviate this problem, the support measure was introduced. Note that $like(first|E)$ is also a measure of support since it is maximal when a value occurs all the time in every example of one class. Hence, adding the support term is just

$$P(first|E)*support(first|E) = \frac{like(first|E)^2}{like(first|E) + like(rest|E)}. \tag{2}$$

To compute the *power* of a subrange, we first apply equal frequency binning (EFB) to each attribute in the dataset. EFB divides the range of possible values into n bins or subranges, each of which holds the same number of attribute values. However, to avoid duplicate values being placed into different bins, boundaries of every pair of neighboring bins are adjusted so that duplicate values should belong to one bin only [9]. For these experiments, we did not optimize the value for n for each dataset. We simply used $n = 10$ bins. In future work, we will dynamically set the value of n for a given dataset.

Algorithm 1. Finding subrange *Power*.

```
1:  Power(D, E) {D is the dataset, and E is a set of
      sub-ranges for a given attribute}
2:  Partition(D) ↦ C {Returns data partitioned according
      to the class label.}
3:  PR ← ∅ {Initialize subranges with power for each
      sub-range in E}
4:  for $j = 0$ to # of class values in $|C|$ do
5:      first ← $C_j$
6:      rest ← $C_{\neq j}$
7:      $p_{first} \leftarrow \frac{|first|}{|D|}$ {Probability of first data}
8:      $p_{rest} \leftarrow \frac{|rest|}{|D|}$ {Probability of rest data}
9:      for $k = 0$ to # of subranges in $|E|$ do
10:         like(first|$E_k$) ← number of times $E_k$ appears in
              first × $p_{first}$
```

```
11:         like(rest|$E_k$) ← number of times $E_k$ appears in
            rest × $p_{rest}$
12:         $power_k \leftarrow \frac{like(first|E_k)^2}{like(first|E_k)+like(rest|E_k)}$
13:         PR ← $power_k$
14:     end for
15: end for
16: return $PR$
```

Next, *CLIFF* selects p *percent* of the rows in a dataset D containing the most powerful subranges. The matrix M holds the result of *Power* for each attribute, for each class in D. This is used to help select the rows from D to produce D' within *CliffSelection*.

The *for*-loop in Lines 3 to 9 of Algorithm 2 iterates through attributes in D and *UniqueRanges* is called to find and return the unique subranges for each attribute. Inside that loop, at Lines 5 to 8, a nested *for*-loop iterates through the unique subranges for a given attribute and *Power* is called to find the power of each subrange. Finally, once the *power*s are found for each attribute subrange, *CliffSelection* is called to determine which rows in D will make up the final sample in D'.

- Partition D by the class label.
- For each row in each partition, find the product of the *power* of the subranges in that row.
- For each partition, return the p percent of the partitioned data with the highest *power*.

An example of how CLIFF is applied to a dataset is described in Section 4.3.

Algorithm 2. The *CLIFF* algorithm.

```
1: CLIFF(D, p) {D is the original dataset, and p is the
   percentage of data to be returned}
2: M ← ∅ {Initialize subrange power for each attribute}
3: for $i = 0$ to # of attributes in D do
4:     UniqueRanges($D_i$)↦$R_i$ {Returns set of unique
       subranges for a given attribute}
5:     for $j = 0$ to # of subranges in $R_i$ do
6:         Power(D, $R_i$) ↦$PR_j$ {Returns the subranges with
           their powers for each class}
7:         M ← $PR_j$
8:     end for
9: end for
10: CliffSelection(D, p, M) ↦$D'$ {Returns p of the original
    data}
11: return $D'$
```

4.2 MORPH

MORPH is an instance mutator that changes the numeric attribute values of each row by replacing these original values with *MORPHed* values. MORPH takes care never to change an instance such that it moves across the boundary between the original instance and instances of another class.

The MORPHed instances are created by applying (3) to each attribute value of the instance

$$y = x \pm (x - z) * r. \tag{3}$$

Let $x \in D$ be the original row to be changed, y the resulting MORPHed row, and $z \in D$ the nearest unlike neighbor (NUN) of x. NUN is the nearest neighbor of x whose class label is different from x's class label (distance is calculated using the *euclidean* distance). The random number r is calculated with the property

$$\alpha \leq r \leq \beta,$$

where $\alpha = 0.15$ and $\beta = 0.35$. The values for α and β are chosen in an effort to keep the MORPHed value close enough to the original to maintain the utility of the dataset but far enough to keep the original data private. Future work will determine the optimal values for these experimental parameters.

A simple hashing scheme lets us check if the new instance y is the same as an existing instance (and we keep MORPHing x until it does not hash to the same value as an existing instance). Hence, we can assert that none of the original instances are found in the final dataset.

An example of how MORPH is applied to a dataset is described in Section 4.3.

4.3 Example of CLIFF+MORPH

For this example, we use the abbreviated version of the *ant-1.3* dataset shown in Table 2a. This dataset contains 10 attributes: one dependent attribute (class) and nine independent attributes. Each row is labeled as 1 (containing at least one defect) or 0 (having no defects). The first column holds the row number and each cell contains the original metric values.

The result of applying CLIFF+MORPH is shown in Table 2e. To get to that point, first the original data is binned using EFB. The result of this is shown in Table 2b. For example, the attribute values of *wmc* are replaced by two subranges of values ([3-6] and (6-14]). Here, all values from 3 to 6 inclusive are placed in the first subrange and all values between 6 and 14 (not including 6) are placed in the last subrange.

Following this, each subrange is ranked according to (2). To find the *power* of each subrange, we first divide the data into *first* and *rest*. For this example, let us say that all the rows with the 0 class label are *first* while the others are *rest*. Fig. 2 shows an example of finding the *power* of (6-14] for attribute *wmc* and Table 2c shows the *power* values for all the subranges of Table 2b.

Next, the *power* of each row is calculated by finding the product of the subrange *powers* of each row. In this example, the row with the highest *power* for each class is selected. In this case that is row 3 for the 0 class label and row 8 for the 1 class label. This result is shown in Table 2d.

Finally, we MORPH this result according to (3) to obtain the result in Table 2e.

5 Evaluation Metrics

This section assesses the privacy and utility offered by CLIFF+MORPH.

5.1 Evaluating Utility via Classification

We say that the classification performance of a privatized dataset is *adequate* if it performs no worse than the *baseline* computed from the original dataset defined as follows:

For datasets in $T, T_1, \ldots, T_{|T|}$, performance data is collected when a defect model is learned from the $All - T_i$, then applied to T_i.

TABLE 2
Example of CLIFF+MORPH: (a) The Original Data and an Abbreviated Version of *ant-1.3*; (b) Data from "a" Binned Using EFB; (c) Power Values for Each Subrange in "b"; (d) CLIFF Result, and (e) MORPH Result

(a) Partial ant-1.3 Defect Data

#	wmc	dit	noc	cbo	rfc	lcom	ca	ce	loc	class
1	11	4	2	14	42	29	2	12	395	0
2	14	1	1	8	32	49	4	4	297	1
3	3	2	0	1	9	0	0	1	58	0
4	12	3	0	12	37	32	0	12	310	0
5	6	3	0	4	21	1	0	4	136	0
6	5	1	5	12	11	8	11	1	59	0
7	4	2	0	3	16	0	0	3	59	0
8	14	1	0	24	63	63	20	20	822	1

(b) ant-1.3 After Equal Frequency Binning

#	wmc	dit	noc	cbo	rfc	lcom	ca	ce	loc	class
1	(6-14]	[1-4]	[0-5]	(8-24]	(21-63]	(8-63]	(2-20]	(4-20]	(136-822]	0
2	(6-14]	[1-4]	[0-5]	[1-8]	(21-63]	(8-63]	(2-20]	[1-4]	(136-822]	1
3	[3-6]	[1-4]	[0-5]	[1-8]	[9-21]	[0-8]	0	[1-4]	[58-136]	0
4	(6-14]	[1-4]	[0-5]	(8-24]	(21-63]	(8-63]	0	(4-20]	(136-822]	0
5	[3-6]	[1-4]	[0-5]	[1-8]	[9-21]	[0-8]	0	[1-4]	[58-136]	0
6	[3-6]	[1-4]	[0-5]	(8-24]	[9-21]	[0-8]	(2-20]	[1-4]	[58-136]	0
7	[3-6]	[1-4]	[0-5]	[1-8]	[9-21]	[0-8]	0	[1-4]	[58-136]	0
8	(6-14]	[1-4]	[0-5]	(8-24]	(21-63]	(8-63]	(2-20]	(4-20]	(136-822]	1

(c) ant-1.3 Power Values

#	wmc	dit	noc	cbo	rfc	lcom	ca	ce	loc	class
1	0.13	0.56	0.60	0.28	0.13	0.13	0.28	0.17	0.28	0
2	0.13	0.06	0.06	0.03	0.13	0.13	0.03	0.03	0.03	1
3	**0.50**	**0.56**	**0.56**	**0.28**	**0.50**	**0.50**	**0.75**	**0.40**	**0.50**	**0**
4	0.13	0.56	0.56	0.28	0.13	0.13	0.75	0.17	0.28	0
5	0.50	0.56	0.56	0.28	0.50	0.50	0.75	0.40	0.50	0
6	0.50	0.56	0.56	0.28	0.50	0.50	0.28	0.40	0.50	0
7	0.50	0.56	0.56	0.28	0.50	0.50	0.75	0.40	0.50	0
8	**0.13**	**0.06**	**0.06**	**0.03**	**0.13**	**0.13**	**0.03**	**0.04**	**0.03**	**1**

(d) CLIFF Result

#	wmc	dit	noc	cbo	rfc	lcom	ca	ce	loc	class
3	3	2	0	1	9	0	0	1	58	0
8	14	1	0	24	63	63	20	20	822	1

(e) MORPH Result

#	wmc	dit	noc	cbo	rfc	lcom	ca	ce	loc	class
3	2	2	0	1	5	5	2	0	1	0
8	15	1	0	26	67	68	22	21	879	1

Table 3 defines performance measures used in this work to assess defect predictors. It also includes a new measure used in this work called the *g*-measure.

- Recall, or *pd*, measures how many of the target (defective) instances are found. The higher the pd, the fewer the false negative results.
- Probability of false alarm, or *pf*, measures how many of the instances that triggered the detector actually did not contained the target (defective) concept. Like pd, the highest pf is 100 percent; however, its desired result is 0 percent.
- *g*-measure (harmonic mean of *pd* and *1-pf*): In this paper, we report on the *g*-measure. The *1-pf* represents *specificity* (not predicting instances without defects as defective). *Specificity* (1-pf) is used together with *pd* to form the *G-mean*$_2$ measure seen in Jiang et al. [33]. It is the geometric mean of the *pd*s for both the majority and the minority class. In our case, we use these to form the *g*-measure, which is the harmonic mean of *pd* and *1-pf*.

Other measures such as *accuracy*, *precision*, and *f-measure* were not used since they are poor indicators of performance for data where the target class is rare (in our case, the *defective* instances). This is based on a study done by Menzies et al. [34] which shows that when datasets contain a low percentage of defects, precision can be unstable. If we look at the datasets in Table 4, we see that the percentage of defects is low in most cases.

$$
\begin{aligned}
\text{E} &= (6\text{ - }14] \\
\text{P(first)} &= {}^{6}/_{8} \\
\text{P(rest)} &= {}^{2}/_{8} \\
\text{freq(E}|\text{first)} &= {}^{2}/_{6} \\
\text{freq(E}|\text{rest)} &= {}^{2}/_{2} \\
\text{like(first}|\text{E)} &= {}^{2}/_{6} \times {}^{6}/_{8} = 0.25 \\
\text{like(rest}|\text{E)} &= {}^{2}/_{2} \times {}^{2}/_{8} = 0.25 \\
\text{P(first}|\text{E)} \times \text{support(first}|\text{E)} &= \frac{0.25^2}{0.25+0.25} = 0.13
\end{aligned}
$$

Fig. 2. Finding the *power* of (6-14].

TABLE 3
Some Popular Measures Used in Software Defect Prediction Work

		Actual	
		yes	no
Predicted	**yes**	TP	FP
	no	FN	TN
recall (pd)	$\frac{TP}{TP+FN}$		
pf	$\frac{FP}{FP+TN}$		
g-measure	$\frac{2*pd*(1-pf)}{pd+(1-pf)}$		

5.2 Evaluating Privatization

Privacy is not a binary step function where something is either 100 percent private or 100 percent disclosed. Rather it is a probabilistic process where we strive to decrease the likelihood that an attacker can uncover something that they should not know. The rest of this section defines privacy using a probabilistic IPR of privatized datasets.

Defining privacy. To investigate how well the original defect data is privatized, we assume the role of an attacker armed with some background knowledge from the original dataset and also supplied with the private dataset. To keep the privatized dataset *truthful*, Brickell and Shmatikov [6] kept the sensitive attribute values as is and privatized only the QIDs. However, in this work, in addition to privatizing the QIDs with CLIFF+MORPH, we apply EFB to the sensitive attribute to create 10 subranges of values to easily report on the privacy level of the privatized dataset.

TABLE 4
Defect Dataset Characteristics

Data	Instances	Attributes	Class	Defect%
ant13	125	20	2	20
arc	234	20	2	12
camel10	339	20	2	4
poi15	237	20	2	41
redaktor	176	20	2	15
skarbonka	45	20	2	20
tomcat	858	20	2	9
velocity14	196	20	2	75
xalan24	723	20	2	15
xerces12	440	20	2	16

TABLE 5
The C-K Metrics of the Datasets Used in This Work (See Table 4)

Attributes	Symbols	Description
average method complexity	amc	e.g., number of JAVA byte codes
average McCabe	avg_cc	average McCabe's cyclomatic complexity seen in class
afferent couplings	ca	how many other classes use the specific class
cohesion amongst classes	cam	summation of number of different types of method parameters in every method divided by a multiplication of number of different method parameter types in whole class and number of methods
coupling between methods	cbm	total number of new/redefined methods to which all the inherited methods are coupled
coupling between objects	cbo	increased when the methods of one class access services of another
efferent couplings	ce	how many other classes is used by the specific class
data access	dam	ratio of the number of private (protected) attributes to the total number of attributes
depth of inheritance tree	dit	provides the position of the class in the inheritance tree
inheritance coupling	ic	number of parent classes to which a given class is coupled (includes counts of methods and variables inherited)
lack of cohesion in methods	lcom	number of pairs of methods that do not share a reference to an instance variable
another lack of cohesion measure	locm3	if m, a are the number of $methods, attributes$ in a class number and $\mu(a)$ is the number of methods accessing an attribute, then $lcom3 = ((\frac{1}{a}\sum_j^a \mu(a_j)) - m)/(1-m)$
lines of code	loc	measures the volume of code
maximum McCabe	max_cc	maximum McCabe's cyclomatic complexity seen in class
functional abstraction	mfa	number of methods inherited by a class plus number of methods accessible by member methods of the class
aggregation	moa	count of the number of data declarations (class fields) whose types are user defined classes
number of children	noc	measures the number of immediate descendants of the class.
number of public methods	npm	counts all the methods in a class that are declared as public. The metric is known also as Class Interface Size (CIS)
response for a class	rfc	number of methods invoked in response to a message to the object
weighted methods per class	wmc	the number of methods in the class (assuming unity weights for all methods).
defects	defects	number of defects per class, seen in post-release bug-tracking systems

The last row is the dependent variable. Jureczko and Madeyski [35] provide more information on these metrics.

We propose the following privacy metric based on the *adversarial accuracy gain, A_{acc},* from the work of Brickell and Shamtikov [6]. According to the authors' definition of A_{acc}, it quantifies an attacker's ability to predict the sensitive attribute value of a target t. The attacker accomplishes this by guessing the most common sensitive attribute value in $\langle t \rangle$ (a QID group).

Specifically, A_{acc} measures the increase in the attacker's accuracy after he observes a privatized dataset and compares it to the baseline from a trivially privatized dataset that offers perfect privacy by removing either all sensitive attribute values or all the other QIDs.

Recall that we assume that the attacker has access to a privatized version (T') of an original dataset (T) and knowledge of nonsensitive QID values for a specific target in T. We refer to the latter as a query. For our experiments, we randomly generate up to 1,000 of these queries, $|Q| \leq$ 1,000 (Section 6.4 describes how queries are generated).

For each query q in a set $Q = \{q_1, \ldots, q_{|Q|}\}$, G_i^* is a group of rows from any dataset which matches q_i. Hence, let G_i be the group from the original dataset and G_i' be the group from the privatized dataset which matches q_i. Next, for every sensitive attribute subrange in the set $S = \{s_1, \ldots, s_{|S|}\}$, we denote the idea of the most common sensitive attribute value as $s_{max}(G_i^*)$.

Now, we define a breach of privacy as follows:

$$Breach(S, G_i^*) = \begin{cases} 1, & if \quad s_{max}(G_i) = s_{max}(G_i'), \\ 0, & otherwise. \end{cases}$$

Therefore, the privacy level of the privatized dataset is

$$100 \times IPR(T^*) = 1 - \frac{1}{|Q|}\sum_{i=0}^{|Q|} Breach(S, G_i^*).$$

$IPR(T^*)$ stands for *IPR* and has some similarity to A_{acc} of Brickell and Shamtikov [6], where $IPR(T^*)$ measures the attacker's ability to cause privacy breaches after observing the privatized dataset T' compared to a baseline of the original dataset T. To be more precise, $IPR(T^*)$ measures the percent of total queries that did not cause a privacy *Breach*.

We baseline our work against the original dataset (our worst-case scenario), which offers no privacy and therefore its $IPR(T) = 0$. In our case, to have perfect privacy (our best-case scenario), we create a privatized dataset by simply removing the sensitive attribute values. This will leave us with $IPR(T') = 1$.

6 Experiments

6.1 Data

To assist replication, our data comes from the online PROMISE data repository [18]. Table 5 describes the attributes of these datasets and Table 4 shows other details.

6.2 Design

From our experiments, the goal is to determine whether we can have effective defect prediction from shared data, while preserving privacy. To test the shared data scenario, we do CCDP experiments for all the datasets of Table 4 in their original state and after they have been privatized. When experimenting with the original data, from the 10 datasets used one is used as a test set while a defect predictor was made from *All* the data in the other nine datasets. This process is repeated for each dataset. The same process is followed when the datasets are privatized. Each of the nine datasets used to create a defect predictor is first privatized, then combined into *All*. The test set is not privatized.

In a separate experiment to test how *private* the datasets are after the privatization algorithms are applied, we model an attacker's background knowledge with queries (see Section 6.4). These queries are applied to both the original and privatized datasets. The *IPR*s for each privatized dataset is then calculated (see Section 5.2).

6.3 Defect Predictors

To analyze the utility of CLIFF+MORPH, we perform CCDP experiments with three classification techniques implemented in WEKA [36]. These are NB [37], SVMs [38], and NNs [39]. The default values for these classifiers in WEKA are used in our experiments.

These three classification techniques have been widely used for defect prediction [11], [40], [41]. Lewis [37] describes NB as a classifier based on Baye's rule. It is a statistical-based learning scheme which assumes that attributes are equally important and statistically independent. To classify an unknown instance, NB chooses the class with the maximum likelihood of containing the evidence in the test case. SVMs seek to minimize misclassification errors by selecting a boundary or hyperplane that leaves the maximum margin between the two classes, where the margin is defined as the sum of the distances of the hyperplane from the closest point of the two classes [42]. According to Lessmann et al. [40], NNs depict a network structure which defines a concatenation of weighting, aggregation, and thresholding functions that a applied to a software module's attributes to obtain an approximation of its posterior probability of being fault prone.

6.4 Query Generator

A *query generator* is used to provide a sample of attacks on the data. Before discussing the query generator, a few details must be established. First, to maintain some "truthfulness" to the data, a selected sensitive attribute and the class attribute are not used as part of query generation. Here, we are assuming that the only information an attacker could have is information about the nonsensitive QIDs in the dataset. As a result, these attribute values (sensitive and class) are unchanged in the privatized dataset.

To illustrate the application of the query generator, we use Tables 2a and 2b. First, EFB is applied to the original data in Table 2a to create the subranges shown in Table 2b. Next, to create a query, we proceed as follows:

1. Given a query size (measured as the number of {attribute subrange} pairs). For this example, we use a query size of 1.
2. Given the set of attributes $A = (\mathrm{wmc, dit, noc, cbo, rfc, lcom, ca, ce})$ and all their possible subranges.
3. Randomly select an attribute from A, for example, *wmc* with two possible subranges (6-14] and [0-6].
4. Randomly select a subrange from all possible subranges of *wmc*, for example, (6-14].

In the end, the query we generate is $wmc = (6-14]$. Table 6 shows more examples of queries, their sizes, the number, and rows they match from the dataset.

For each query size, we generate up to 1,000 queries because it is not practical to test every possible query. With

TABLE 6
Example: Queries, Query Sizes and the Number of Rows That Match the Queries, $|G|$

Query	Size	|G|	Row#
cbo = (8-24]	1	4	1,4,6,8
cbo = (8-24] and wmc = (6-14]	2	3	1,4,8
cbo = (8-24] and wmc = (6-14] and noc = [0-5] and ca = 0	4	1	4

Table 2b is used for this example.

these datasets the number of possible queries with arity 4 and no repeats is 38,760,000.[3]

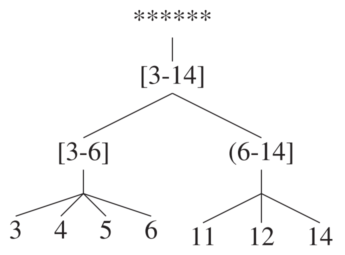
Each query must also satisfy the following *sanity checks*:

- They must not include attribute value pairs from either the designated sensitive attribute or the class attribute.
- They must return at least two instances after a search of the original dataset.
- They must not be the same as another query no matter the order of the individual {attribute subrange} pairs in the query.

6.5 Benchmark Privacy Algorithms

To benchmark our approach, we need to compare it against data swapping and *k*-anonymity. Data swapping is a standard perturbation technique used for privacy [19], [43], [44]. This is a permutation approach that deassociates the relationship between a nonsensitive QID and a numerical sensitive attribute. In our implementation of data swapping, for each QID a certain percent of the values are swapped with any other value in that QID. For our experiments, these percentages are 10, 20, and 40 percent.

An example of *k*-anonymity is shown in Fig. 1. Our implementation follows the Datafly algorithm [45] for *k*-anonymity. The core Datafly algorithm starts with the input of a set of QIDs, *k*, and a generalization hierarchy. An example of a hierarchy is shown in the tree below using the values of *wmc* from Table 2a. Values at the leaves are generalized by replacing them with the subranges [3-6] or (6-14]. These in turn can be replaced by [3-14]. Or the leaf values can be suppressed by replacing them with a symbol such as the *stars* at the top of the tree.



Datafly then replaces values in the QIDs according to the hierarchy. This generalization continues until there are *k* or fewer distinct instances. These instances are suppressed.

3. $\frac{n!}{k!(n-k)!} = \binom{n}{k}$, where n is 19 (all attributes minus the class and sensitive attributes), and $k = 4$. This gives 3,876 combinations. A combination (size 4 query) over variables with 10 values each (i.e., the 10 values generated by EFB) generates a space of 10^4 options. Therefore, the total number of possible queries of arity four is $10^4 * 3876 = 38{,}760{,}000$.

TABLE 7
Algorithm Characteristics

Algorithms	Symbol	Meaning
MORPHed	m	data privatized by MORPH only
MORPHed$X$	mX: m10, m20, m40	X represents the percentage of the original data that remains after CLIFF is applied and m indicates that the remaining data is then MORPHed. These are collectively referred to as CLIFF+MORPH algorithms.
swapped$X$	sX: s10, s20, s40	X represents the percentage of the original data swapped.
X-anonymity	kX: k2, k4	X represents the size of the QID group, where each member of the group is indistinguishable from k - 1 others in the group.

6.6 Experimental Evaluation

This section presents the results of our experiments. Before going forward, Table 7 shows the notation and meaning of the algorithms used in this work. The results are structured according to the research questions presented in Section 1.

RQ1: Does CLIFF+MORPH provide better balance between privacy and utility than other state-of-the-art privacy algorithms?

To see whether CLIFF+MORPH offers a better balance between privacy and utility, we privatized the original datasets shown in Table 4 with data swapping, k-anonymity, and CLIFF+MORPH. Then, the IPR and g-measures are calculated for each privatized dataset. The experimental results are displayed in Fig. 3. For each chart, we plot IPR on the x-axis and g-measures on the y-axis. The g-measures are based on the NB and the IPR based on queries of size 1. There is not enough space in this paper to repeat Fig. 3 for each query size and each classifier used in this work (a total of nine figures). Instead, we present IPR results for query sizes 2 and 4 in Table 8, and g-measures for NB, SVMs, and NNs in Tables 9, 10, and 11, respectively.

In Fig. 3, the *horizontal* and *vertical* lines show the CCDP g-measures of the original dataset and $IPR = 80\%$, respectively. To answer RQ1, we say that the privatized datasets appearing to the *right* and *above* these lines, are *private enough* (IPR over 80 percent) and *performs adequately* (as good as or better than the nonprivatized data). In other words, this region corresponds to data that provide the best balance between privacy and utility.

Note that for most cases, $\frac{7}{10}$, the CLIFF+MORPH algorithms (m10, m20, and m40) fall to the *right* and *above* these lines. We expect this result for IPRs; however, one may question why the g-measures are higher than those of the original dataset in those seven cases. This is due to instance selection done by CLIFF. Research has shown that instance selection can produce comparable or improved classification results [31], [46]. This makes CLIFF a perfect compliment to MORPH. Other works on instance selection are included in the surveys, [47], [48], [49], [50].

Of the three remaining datasets (camel-1.0, tomcat, and xalan-2.4), all offer IPRs above the 80 percent baseline; however, their g-measures are either as good as those of the original dataset or worse. This is particularly true in the case of xalan-2.4. We see this as an issue concerning the structure of the dataset rather than the CLIFF+MORPH algorithms themselves, considering that it performs well for most of the privatized datasets used in this study.

Equally important is the fact that, generally, the data-swapping algorithms are *less* private than the CLIFF+MORPH algorithms and k-anonymity. This effect is seen clearly in Fig. 3 where, all 10 datasets show the best IPRs belonging to the CLIFF+MORPH algorithms and k-anonymity. While the utility of data swapping and k-anonymity are generally as good as or worse than the nonprivatized dataset.

This summary of results also holds true when measuring IPRs for query sizes 2 and 4 (see Table 8) and g-measures for SVMs (see Table 10) and NNs (see Table 11).

So, to answer RQ1, at least for the defect datasets used in this study and the baselines used, CLIFF+MORPH generally provides a better balance between privacy and utility than data swapping and k-anonymity.

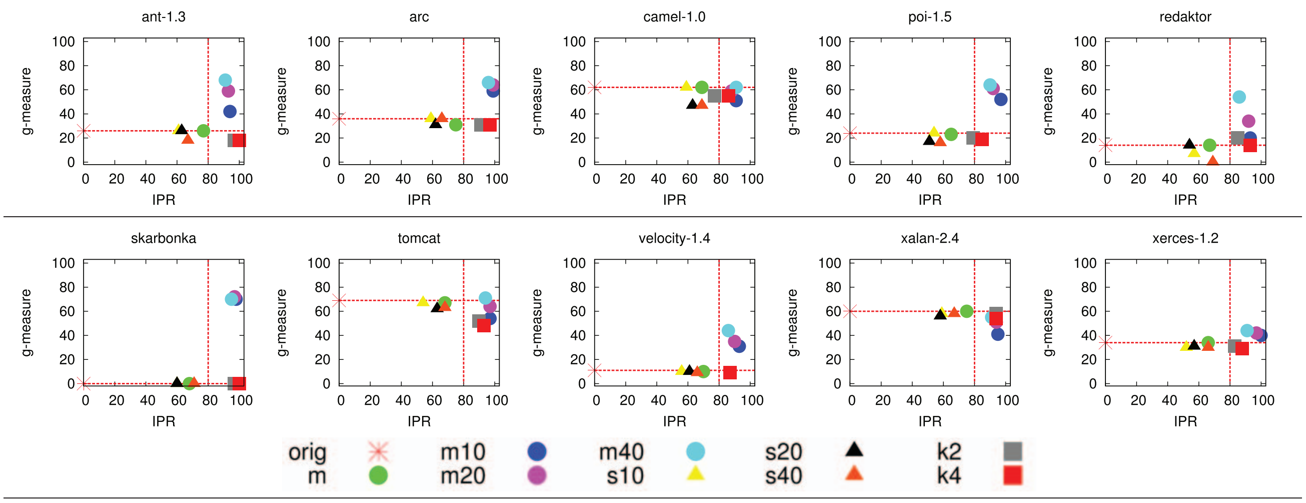


Fig. 3. G-Measure versus IPR with query size 1 for all 10 datasets. The *horizontal* and *vertical* lines show the CCDP g-measures from the NB defect model and $IPR = 80\%$ (respectively). Note that points above and to the right of these lines are *private enough* (IPR over 80 percent) and *performs adequately* (as good as or better than the original data).

TABLE 8
IPR for Query Sizes 2 and 4

Query Size = 2

Data	**m**	**m10**	**m20**	**m40**	**s10**	**s20**	**s40**	**k2**	**k4**
ant-1.3	77.1	**97.3**	95.9	91.9	60.2	67.6	**78.6**	98.4	**99.9**
arc	75.4	**99.4**	98.3	96.8	60.7	64.7	**79.0**	94.0	**99.2**
camel-1.0	80.3	**95.6**	94.1	93.9	62.3	68.2	**78.7**	89.9	**94.7**
poi-1.5	76.2	**97.9**	97.1	94.6	60.5	66.9	**74.1**	90.7	**95.3**
redaktor	75.3	**94.9**	93.7	90.8	61.3	64.2	**76.2**	89.3	**97.6**
skarbonka	77.5	**99.7**	97.9	96.1	57.5	65.0	**76.6**	98.8	**100.0**
tomcat	79.6	**98.2**	96.0	90.6	64.0	69.6	**73.9**	92.9	**96.6**
velocity-1.4	76.6	**96.0**	94.6	90.5	60.1	65.8	**75.7**	92.9	**95.4**
xalan-2.4	79.6	**96.5**	93.9	91.8	63.6	67.8	**76.3**	93.4	**97.0**
xerces-1.2	76.7	**100.0**	99.2	96.9	58.3	64.1	**72.6**	89.6	**95.1**
MEDIAN	76.9	**97.6**	96.0	92.9	60.6	66.4	**76.3**	92.9	**96.8**

Query Size = 4

Data	**m**	**m10**	**m20**	**m40**	**s10**	**s20**	**s40**	**k2**	**k4**
ant-1.3	78.6	**97.5**	97.0	95.4	62.2	68.5	**78.3**	99.6	**99.6**
arc	78.0	**99.5**	98.6	98.2	59.3	69.2	**77.7**	99.2	**100.0**
camel-1.0	75.0	**81.1**	80.4	80.0	57.9	67.3	**72.8**	80.0	**80.8**
poi-1.5	78.0	**99.8**	99.5	98.9	60.8	70.6	**77.9**	97.5	**98.4**
redaktor	81.6	**98.8**	97.6	95.9	60.2	68.0	**77.5**	95.8	**100.0**
skarbonka	61.5	**100.0**	100.0	98.9	56.2	59.0	**64.2**	100.0	**100.0**
tomcat	84.3	**99.7**	99.5	98.8	61.3	72.3	**85.0**	99.1	**99.8**
velocity-1.4	77.8	**100.0**	98.7	98.1	59.8	67.8	**77.7**	99.1	**99.5**
xalan-2.4	81.1	**99.9**	99.1	98.2	60.4	69.7	**80.0**	99.2	**100.0**
xerces-1.2	78.3	**100.0**	100.0	100.0	59.8	65.5	**77.6**	99.0	**99.5**
MEDIAN	78.2	**99.8**	98.9	98.2	60.0	68.3	**77.7**	99.1	**99.7**

The numbers in bold are the highest for the different types of privacy algorithms.

TABLE 10
Cross-Company Experiment for 10 Datasets with SVMs: The *pd*s, *pf*s, and g-Measures Are Shown with the g-Measures Shown in Bold

SVM		**orig**	**m**	**m10**	**m20**	**m40**	**s10**	**s20**	**s40**	**k2**	**k4**
ant1.3	***g***	**0**	**0**	**62**	**75**	**75**	**0**	**0**	**0**	**0**	**0**
	pd	*0*	*0*	*90*	*70*	*70*	*0*	*0*	*0*	*0*	*0*
	pf	*0*	*0*	*52*	*20*	*18*	*0*	*0*	*0*	*0*	*0*
arc	***g***	**0**	**0**	**67**	**59**	**56**	**0**	**0**	**0**	**0**	**0**
	pd	*0*	*0*	*70*	*44*	*41*	*0*	*0*	*0*	*0*	*0*
	pf	*0*	*0*	*36*	*14*	*9*	*0*	*0*	*0*	*0*	*0*
camel1.0	***g***	**0**	**0**	**70**	**71**	**67**	**0**	**0**	**0**	**0**	**0**
	pd	*0*	*0*	*85*	*62*	*54*	*0*	*0*	*0*	*0*	*0*
	pf	*0*	*0*	*41*	*16*	*13*	*0*	*0*	*0*	*0*	*0*
poi1.5	***g***	**0**	**0**	**54**	**25**	**30**	**0**	**0**	**0**	**0**	**0**
	pd	*0*	*0*	*41*	*14*	*18*	*0*	*0*	*0*	*0*	*0*
	pf	*0*	*0*	*20*	*2*	*4*	*0*	*0*	*0*	*0*	*0*
redaktor	***g***	**0**	**0**	**40**	**50**	**55**	**0**	**0**	**0**	**0**	**0**
	pd	*0*	*0*	*63*	*59*	*56*	*0*	*0*	*0*	*0*	*0*
	pf	*0*	*0*	*70*	*56*	*46*	*0*	*0*	*0*	*0*	*0*
skarbonka	***g***	**0**	**0**	**68**	**48**	**35**	**0**	**0**	**0**	**0**	**0**
	pd	*0*	*0*	*67*	*33*	*22*	*0*	*0*	*0*	*0*	*0*
	pf	*0*	*0*	*31*	*14*	*14*	*0*	*0*	*0*	*0*	*0*
tomcat	***g***	**0**	**0**	**60**	**73**	**65**	**0**	**0**	**0**	**0**	**5**
	pd	*0*	*0*	*90*	*65*	*51*	*0*	*0*	*0*	*0*	*3*
	pf	*0*	*0*	*54*	*16*	*9*	*0*	*0*	*0*	*0*	*0*
velocity1.4	***g***	**0**	**0**	**43**	**27**	**18**	**0**	**0**	**0**	**0**	**0**
	pd	*0*	*0*	*44*	*16*	*10*	*0*	*0*	*0*	*0*	*0*
	pf	*0*	*0*	*57*	*16*	*10*	*0*	*0*	*0*	*0*	*0*
xalan2.4	***g***	**0**	**0**	**66**	**61**	**62**	**0**	**0**	**0**	**0**	**24**
	pd	*0*	*0*	*70*	*56*	*63*	*0*	*0*	*0*	*0*	*14*
	pf	*0*	*0*	*37*	*34*	*38*	*0*	*0*	*0*	*0*	*2*
xerces1.2	***g***	**0**	**0**	**46**	**48**	**42**	**0**	**0**	**0**	**0**	**0**
	pd	*0*	*0*	*45*	*39*	*28*	*0*	*0*	*0*	*0*	*0*
	pf	*0*	*0*	*52*	*38*	*14*	*0*	*0*	*0*	*0*	*0*
MEDIAN (g)		0	0	**61**	**54**	**55**	0	0	0	0	0

RQ2: Do different classifiers affect the experimental results?

Looking at the raw results from Tables 9, 10, and 11, the SVM results shown in Table 10 standout for CLIFF+MORPH and the other privacy algorithms. Here, in all cases CLIFF+MORPH has better g-measures, while in $\frac{8}{10}$ cases, only CLIFF+MORPH has g-measures above zero. The privacy algorithms for NB and the NN generally show better g-measure results than SVM.

For a substantial comparison between classifiers, we use the Mann-Whitney U Test [51] at 95 percent confidence for the g-measures over all datasets. The results show that there is no significant difference in the performance of NB and NNs. However, when g-measures from SVM are compared to those of NB and NNs, it performs significantly worse. Therefore, to answer RQ2, although the results show that the NB and NN classifiers perform better

TABLE 9
Cross-Company Experiment for 10 Datasets with NB: The *pd*s, *pf*s, and g-measures Are Shown with the g-Measures Shown in Bold

NB		**orig**	**m**	**m10**	**m20**	**m40**	**s10**	**s20**	**s40**	**k2**	**k4**
ant1.3	***g***	**26**	**26**	**42**	**59**	**68**	**26**	**26**	**18**	**18**	**18**
	pd	*15*	*15*	*95*	*85*	*90*	*15*	*15*	*10*	*10*	*10*
	pf	*5*	*7*	*73*	*55*	*46*	*5*	*5*	*5*	*7*	*7*
arc	***g***	**36**	**31**	**59**	**64**	**66**	**36**	**31**	**36**	**31**	**31**
	pd	*22*	*19*	*81*	*67*	*59*	*22*	*19*	*22*	*19*	*19*
	pf	*5*	*5*	*54*	*38*	*26*	*5*	*5*	*5*	*5*	*5*
camel1.0	***g***	**62**	**62**	**51**	**59**	**62**	**62**	**47**	**47**	**55**	**55**
	pd	*46*	*46*	*92*	*77*	*62*	*46*	*31*	*31*	*38*	*38*
	pf	*5*	*5*	*64*	*52*	*38*	*3*	*4*	*4*	*4*	*4*
poi1.5	***g***	**24**	**23**	**52**	**61**	**64**	**24**	**17**	**16**	**20**	**19**
	pd	*13*	*13*	*84*	*78*	*54*	*13*	*9*	*9*	*11*	*11*
	pf	*4*	*4*	*63*	*50*	*22*	*4*	*3*	*3*	*4*	*4*
redaktor	***g***	**14**	**14**	**20**	**34**	**54**	**7**	**14**	**0**	**20**	**14**
	pd	*7*	*7*	*96*	*89*	*78*	*4*	*7*	*0*	*11*	*7*
	pf	*7*	*5*	*89*	*79*	*59*	*7*	*7*	*3*	*7*	*6*
skarbonka	***g***	**0**	**0**	**70**	**72**	**70**	**0**	**0**	**0**	**0**	**0**
	pd	*0*	*0*	*89*	*89*	*78*	*0*	*0*	*0*	*0*	*0*
	pf	*8*	*8*	*42*	*39*	*36*	*8*	*3*	*3*	*3*	*3*
tomcat	***g***	**69**	**67**	**54**	**64**	**71**	**67**	**62**	**63**	**52**	**48**
	pd	*57*	*55*	*92*	*88*	*81*	*55*	*48*	*48*	*36*	*32*
	pf	*12*	*12*	*62*	*50*	*36*	*12*	*11*	*10*	*9*	*8*
velocity1.4	***g***	**11**	**10**	**31**	**35**	**44**	**10**	**10**	**9**	**9**	**9**
	pd	*6*	*5*	*64*	*41*	*35*	*5*	*5*	*5*	*5*	*5*
	pf	*12*	*12*	*80*	*69*	*41*	*8*	*8*	*8*	*10*	*10*
xalan2.4	***g***	**60**	**60**	**41**	**51**	**55**	**58**	**56**	**58**	**58**	**54**
	pd	*49*	*48*	*92*	*89*	*88*	*47*	*44*	*45*	*47*	*42*
	pf	*24*	*22*	*74*	*64*	*60*	*23*	*22*	*20*	*24*	*23*
xerces1.2	***g***	**34**	**34**	**40**	**42**	**44**	**30**	**31**	**30**	**31**	**29**
	pd	*21*	*21*	*51*	*42*	*34*	*18*	*18*	*18*	*18*	*17*
	pf	*9*	*9*	*67*	*59*	*36*	*9*	*8*	*10*	*7*	*7*
MEDIAN (g)		30	28	**47**	**59**	**63**	28	28	24	25	24

TABLE 11
Cross-Company Experiment for 10 Datasets with NNs: The *pd*s, *pf*s, and g-Measures Are Shown with the g-Measures Shown in Bold

NN		**orig**	**m**	**m10**	**m20**	**m40**	**s10**	**s20**	**s40**	**k2**	**k4**
ant1.3	***g***	**38**	**32**	**59**	**56**	**62**	**38**	**51**	**44**	**32**	**0**
	pd	*25*	*20*	*70*	*65*	*65*	*25*	*35*	*30*	*20*	*0*
	pf	*17*	*18*	*50*	*51*	*41*	*20*	*7*	*15*	*20*	*7*
arc	***g***	**57**	**19**	**60**	**60**	**63**	**43**	**42**	**47**	**68**	**0**
	pd	*44*	*11*	*59*	*59*	*74*	*30*	*30*	*33*	*59*	*0*
	pf	*22*	*22*	*40*	*40*	*45*	*20*	*26*	*20*	*20*	*7*
camel1.0	***g***	**45**	**57**	**65**	**59**	**52**	**52**	**51**	**58**	**44**	**0**
	pd	*31*	*46*	*69*	*62*	*46*	*38*	*38*	*46*	*31*	*0*
	pf	*19*	*25*	*38*	*43*	*41*	*22*	*24*	*21*	*24*	*6*
poi1.5	***g***	**37**	**20**	**45**	**49**	**53**	**33**	**18**	**19**	**42**	**7**
	pd	*23*	*11*	*31*	*36*	*40*	*21*	*10*	*11*	*29*	*4*
	pf	*16*	*16*	*20*	*25*	*25*	*10*	*9*	*8*	*23*	*6*
redaktor	***g***	**43**	**39**	**56**	**63**	**51**	**29**	**48**	**20**	**67**	**14**
	pd	*30*	*26*	*74*	*78*	*70*	*19*	*33*	*11*	*59*	*7*
	pf	*23*	*23*	*54*	*48*	*60*	*38*	*17*	*17*	*24*	*1*
skarbonka	***g***	**20**	**33**	**46**	**47**	**63**	**0**	**20**	**20**	**20**	**0**
	pd	*11*	*22*	*44*	*44*	*56*	*0*	*11*	*11*	*11*	*0*
	pf	*19*	*36*	*53*	*50*	*28*	*14*	*11*	*8*	*11*	*0*
tomcat	***g***	**39**	**38**	**58**	**69**	**68**	**44**	**42**	**46**	**41**	**3**
	pd	*26*	*25*	*52*	*73*	*68*	*30*	*29*	*32*	*29*	*1*
	pf	*20*	*20*	*34*	*34*	*31*	*20*	*18*	*19*	*25*	*8*
velocity1.4	***g***	**14**	**15**	**35**	**35**	**37**	**9**	**13**	**15**	**19**	**5**
	pd	*7*	*8*	*34*	*37*	*29*	*5*	*7*	*8*	*11*	*3*
	pf	*4*	*12*	*63*	*67*	*47*	*20*	*4*	*10*	*14*	*0*
xalan2.4	***g***	**41**	**44**	**58**	**57**	**62**	**35**	**35**	**39**	**36**	**13**
	pd	*27*	*30*	*64*	*59*	*65*	*22*	*22*	*25*	*23*	*7*
	pf	*17*	*20*	*46*	*46*	*41*	*13*	*13*	*17*	*18*	*11*
xerces1.2	***g***	**18**	**28**	**52**	**47**	**45**	**13**	**26**	**13**	**18**	**16**
	pd	*10*	*17*	*56*	*35*	*32*	*7*	*15*	*7*	*10*	*8*
	pf	*9*	*11*	*51*	*31*	*24*	*11*	*11*	*11*	*14*	*6*
MEDIAN (g)		39	33	**57**	**56**	**57**	34	39	29	38	4

than SVM, the *g*-measures show that the CLIFF+MORPH algorithms have the best performance for all three classifiers used in this study. Therefore, based on this result for these different types of classifiers we conclude that the performance of CLIFF+MORPH is not dependent on the type of classifier used.

RQ3: Does CLIFF+MORPH perform better than MORPH?

Previous work by Peters and Menzies [7] showed that a dataset privatized by MORPH offered at least four times more privacy than the original dataset and comparable utility results. In this paper, we enhance the performance of the MORPH algorithm by first applying the instance selector CLIFF to the original dataset.

All results in this study indicate that CLIFF+MORPH performs better than just MORPH. This is because as CLIFF removes the instances with the fewest *power subranges*, it is getting rid of those instances which may cause ambiguous classification. This improves utility. An exceptional example of this can be seen in Fig. 3 with *skarbonka*. When NB is use to create the defect predictor, we see much higher *g*-measures for the CLIFF+MORPH algorithms (between 60 and 80 percent), while the MORPH algorithm has a 0 percent *g*-measure. This general trend of the CLIFF+MORPH algorithms having better utility than the MORPH algorithm can also be seen in Fig. 3 for *ant-1.3*, *arc*, *poi-1.5*, *redaktor*, *velocity-1.4*, and *xerces-1.2*. Additionally, Tables 10 and 11 confirm these results with both SVM and NNs having greater median *g*-measures for the CLIFF+MORPH algorithms than for the MORPH algorithm. The best performance is shown with SVM, where MORPH has a 0 percent median *g*-measure and *m10* has a median 61 percent *g*-measure.

Privacy is improved because we are first removing 90, 80, and 60 percent of the original data, then MORPHing the remaining instances. The removed instances are guaranteed 100 percent protection due to their absence. By then applying MORPH to the remaining instances, we have (in all cases) IPRs for the CLIFF+MORPH algorithms that are *private enough* (>80 percent) and greater than those for MORPH which in most cases are not >80 percent. These results are shown in Fig. 3 and Table 8.

7 Threats to Validity

As with any empirical study, biases can affect the final results. Therefore, any conclusions made from this work must be considered with the following issues in mind:

1. *Sampling bias*. Threatens any classification experiment, i.e., what matters there may not be true here. For example, the datasets used here comes from the PROMISE repository and were supplied by one individual. The best we can do is define our methods and publicize our data so that other researchers can try to repeat our results and, perhaps, point out a previously unknown bias in our analysis. Hopefully, other researchers will emulate our methods to repeat, refute, or improve our results.
2. *Learner bias.* Another source of bias in this study is the learners used for the defect prediction studies. Classification is a large and active field and any single study can only use a small subset of the known classification algorithms. In this work, only results for NB, SVMs, and NNs are published.
3. *Evaluation bias.* This paper has focused on background knowledge specific to the original datasets without regard for other types of background knowledge which cannot be captured by the queries used in this study, for instance, correlation knowledge and knowledge about knowing information about related files. This is a subject for future work.
4. *Other evaluation bias.* The utility of a privatized dataset can be measured semantically (where the workload is unknown) or empirically (known workload, e.g., classification or aggregate query answering). In this paper, we measure utility empirically for defect prediction.
5. *Comparison bias.* There are many anonymization algorithms and it would be difficult to compare the performance of CLIFF+MORPH against all of them. This paper compares our approach against privatization methods that are known *not* to damage classification; this is why we used the data swapping (also used by Taneja et al. [43]). We also used *k*-anonymity [45], a widely used privacy algorithm.

8 Related Work

In this section, we address the two major components of privacy research: 1) the privacy algorithms and 2) the models used to measure privacy levels of these algorithms. We also highlight privacy-preserving data mining, a closely related field in privacy research and noise reduction, as a side effect of the work done in this paper.

8.1 Privacy Research in Software Engineering

To the best of our knowledge, this is the first published result where privatization does not compromise defect prediction. However, it is closely related to privacy work for software testing and debugging [43], [52], [53], [54]. Here, although the work uses within company data, privacy becomes an issue when it involves outsourcing the testing to third parties, as is the case with Budi et al. [54] and Taneja et al. [43], or collecting user information after a software system has been deployed [52], [53]. In the former case, since companies do not wish to release actual cases for testing, they anonymize the test cases before releasing them to testers. In this situation, if the test cases are not able to find the bugs like the original data, then the field of outsourced testing is in danger. Similarly, CCDP can suffer the same fate.

Work published by Castro et al. [52] sought to provide a solution to the problem of software vendors who need to include sensitive user information in error reports to reproduce a bug. To protect sensitive user information, Castro et al. [52] used symbolic execution along the path followed by a failed execution to compute path conditions. Their goal was to compute new input values unrelated to the original input. These new input values satisfied the path conditions required to make the software follow the same execution path until it failed.

As a follow-up to the Castro et al. [52] paper, Clause and Orso [53] presented an algorithm which anonymized input sent from users to developers for debugging. Like Castro et al. [52], the aim of Clause and Orso was to supply the developer with anonymized input which causes the same failure as the original input. To accomplish this, they first use a novel "path condition relaxation" technique to relax the constraints in path conditions, thereby increasing the number of solutions for computed conditions.

In contrast to the work done by Castro et al. [52] and Clause and Orso [53], Taneja et al. [43] proposed PRIEST, a privacy framework. Unlike our work, which privatizes data randomly within NUN border constraints, the privacy algorithm in PRIEST is based on data swapping, where each value in a dataset is replaced by another distinct value of the same attribute. This is done according to some probability that the original value will remain unchanged. An additional difference is in the privacy metric used. They make use of a "guessing anonymity" technique that generates a similarity matrix between the original and privatized data. The values in this matrix are then used to calculate three privacy metrics: 1) mean guessing anonymity, 2) fraction of records with a guessing anonymity greater than $m = 1$, and 3) unique records which determine if any records from the original data remain after privatization.

Work by Taneja et al. [43] followed work done by Budi et al. [54]. Similarly, their work focused on providing privatized data for testing and debugging. They were able to accomplish this with a novel privacy algorithm called kb-anonymity. This algorithm combined k-anonymity with the concept of program behavior preservation, which guide the generation of new test cases based on known ones and make sure the new test cases satisfy certain properties [54]. The difference with the follow-up work by Taneja et al. [43], is that while Budi et al. [54] replace the original data with new data, in Taneja's work [43] the data-swapping algorithm maintains the original data and offers individual privacy by swapping values.

8.2 Privacy Preserving Data Mining

Privacy-preserving data mining is a closely related area of study for data privacy. Developing techniques that can incorporate the privacy concern is an important area in data mining research [55]. In their work, Agrawal and Srikant [55] sought to answer the question of whether accurate models can be developed without access to the precise information in individual data records. In other words, can accurate models be built with privatized datasets. Agrawal and Srikant [55] answered this question by reconstructing an estimation of the distribution of the original data. They showed that they were able to build classifiers whose accuracies were comparable to the accuracies of classifiers built with the original data.

8.3 Noise Detection

This work has a side effect of *noise reduction*. In other work, we have explored the effect of noise on defect data [31], [56]. Kim et al. [56] proposed a closest list noise identification (CLNI) algorithm to reduce noise. To find noisy instances, CLNI first for each instance finds its distance from all other instances. These are then sorted in ascending order and the percentage of top N instances with a different class label from the instance is recorded. If this value is greater than a certain threshold, then the instance has a high probability of being a noisy instance.

9 Conclusion

Studies have shown that early detection and fixing of defects in software projects is less expensive than finding defects later on [57], [58]. Organizations with local data can take full advantage of this early detection benefit by doing WCDP. When an organization does not have enough local data to build defect predictors, they might try to access relevant data from other organizations to perform CCDP. That access will be denied unless the privacy concerns of the data owners can be addressed. Current research in privacy seek to address one issue, i.e., providing adequate privacy for data while maintaining the efficacy of the data. However, reaching an adequate balance between privacy and efficacy has proven to be a challenge since intuitively—the more the data is privatized the less useful the data becomes.

To address this issue we present CLIFF+MORPH, a pair of privacy algorithms designed to privatize defect datasets for CCDP. The data is privatized in two steps: 1) instance pruning with CLIFF, where CLIFF gets rid of irrelevant instances thereby increasing the distances between the remaining instances, and 2) perturbation (synthetic data generation) with MORPH, where MORPH is able to move the remaining instances further and create new synthetic instances that do not cross class boundaries.

Unlike previous studies, we show in Fig. 3 that CLIFF+MORPH 1) maintains increasing data privacy of datasets, 2) without damaging the usefulness of the data for defect prediction. Note that this is a significant result since prior work with the standard privatization technologies could not achieve those two goals.

We hope that this result encourages more data sharing, more cross-company experiments, and more work on building SE models that are general to large-scale systems.

Our results suggest the following future work:

- The experiments of this paper should be repeated on additional datasets.
- The current NUN algorithm used in MORPH is $O(N^2)$. We are exploring ways to optimize that with some clustering index method (e.g., k-means).
- While Fig. 3 shows that we can increase privacy, it also shows that we cannot 100 percent guarantee it. At this time, we do not know the exact levels of privacy required in industry or if the results of Fig. 3 meet those needs. This requires further investigation.
- Currently, we use the MORPH algorithm with set experimental parameter values (α and β). Further investigation is needed to determine the optimal values for these parameters.
- This work focused on showing how to prevent an attacker from associating a target t to a single sensitive attribute value. We are exploring ways to show how a dataset with multiple sensitive attributes can be adequately privatized.
- In the study of data privacy, modeling the attacker's background knowledge is important to determine how private a dataset is. In this paper, we only focused on background knowledge specific to the original datasets. Other types of background knowledge need to be considered.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers whose comments helped them improve the quality of this paper. The work was funded by US National Science Foundation grant CCF:1017330 and the Qatar/West Virginia University grant NPRP 09-12-5-2-470.

REFERENCES

[1] E. Kocaguneli and T. Menzies, "How to Find Relevant Data for Effort Estimation?" *Proc. Int'l Symp. Empirical Software Eng. and Measurement,* pp. 255-264, 2011.

[2] B. Turhan, T. Menzies, A. Bener, and J.D. Stefano, "On the Relative Value of Cross-Company and Within-Company Data for Defect Prediction," *Empirical Software Eng.,* vol. 14, pp. 540-578, 2009.

[3] E. Kocaguneli, G. Gay, T. Menzies, Y. Yang, and J.W.J. Keung, "When to Use Data from Other Projects for Effort Estimation," *Proc. IEEE/ACM Int'l Conf. Automated Software Eng.*, pp. 321-324, 2010.

[4] E. Weyuker, T. Ostrand, and R. Bell, "Do Too Many Cooks Spoil the Broth? Using the Number of Developers to Enhance Defect Prediction Models," *Empirical Software Eng.*, vol. 13, pp. 539-559, Oct. 2008.

[5] M. Grechanik, C. Csallner, C. Fu, and Q. Xie, "Is Data Privacy Always Good for Software Testing?" *Proc. 21st IEEE Int'l Symp. Software Reliability Eng.*, pp. 368-377, 2010.

[6] J. Brickell and V. Shmatikov, "The Cost of Privacy: Destruction of Data-Mining Utility in Anonymized Data Publishing," *Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* pp. 70-78, 2008.

[7] F. Peters and T. Menzies, "Privacy and Utility for Defect Prediction: Experiments with MORPH," *Proc. 34th Int'l Conf. Software Eng.*, pp. 189-199, June 2012.

[8] R. Reed, "Pruning Algorithms-A Survey," *IEEE Trans. Neural Networks,* vol. 4, no. 5, pp. 740-747, Sept. 1993.

[9] S. Kotsiantis and D. Kanellopoulos, "Discretization Techniques: A Recent Survey," *GESTS Int'l Trans. Computer Science and Eng.,* vol. 32, no. 1, pp. 47-58, 2006.

[10] T.J.T. Ostrand, E.J.E. Weyuker, and R.M.R. Bell, "Where the Bugs Are," *Proc. ACM SIGSOFT Int'l Symp. Software Testing and Analysis,* pp. 86-96, 2004.

[11] T. Menzies, J. Greenwald, and A. Frank, "Data Mining Static Code Attributes to Learn Defect Predictors," *IEEE Trans. Software Eng.,* vol. 33, no. 1, pp. 2-13, Jan. 2007.

[12] T. Menzies, Z. Milton, B. Turhan, B. Cukic, Y. Jiang, and A. Bener, "Defect Prediction from Static Code Features: Current Results, Limitations, New Approaches," *Automated Software Eng.,* vol. 17, pp. 375-407, 2010.

[13] A. Tosun, B. Turhan, and A. Bener, "Practical Considerations in Deploying AI for Defect Prediction: A Case Study within the Turkish Telecommunication Industry," *Proc. Fifth Int'l Conf. Predictor Models in Software Eng.*, pp. 11:1-11:9, 2009.

[14] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-Project Defect Prediction: A Large Scale Experiment on Data versus Domain versus Process," *Proc. Seventh Joint Meeting of the European Software Eng. Conf. and the ACM SIGSOFT Symp. Foundations of Software Eng.*, pp. 91-100, 2009.

[15] B.A. Kitchenham, E. Mendes, and G.H. Travassos, "Cross versus Within-Company Cost Estimation Studies: A Systematic Review," *IEEE Trans. Software Eng.*, vol. 33, no. 5, pp. 316-329, May 2007.

[16] Health Information Privacy, http://www.hhs.gov/ocr/privacy/, 2007.

[17] L. Sweeney, "K-Anonymity: A Model for Protecting Privacy," *Int'l J. Uncertainty, Fuzziness and Knowledge-Based Systems,* vol. 10, no. 5, pp. 557-570, 2002.

[18] T. Menzies, B. Caglayan, E. Kocaguneli, J. Krall, F. Peters, and B. Turhan, The Promise Repository of Empirical Software Engineering Data, promisedata.googlecode.com, June 2012.

[19] B.C.M. Fung, R. Chen, and P.S. Yu, "Privacy-Preserving Data Publishing: A Survey on Recent Developments," *Computing,* vol. V, no. 4, pp. 1-53, 2010.

[20] D. Zhu, X.-B. Li, and S. Wu, "Identity Disclosure Protection: A Data Reconstruction Approach for Privacy-Preserving Data Mining," *Decision Support Systems,* vol. 48, no. 1, pp. 133-140, Dec. 2009.

[21] V. Verykios, E. Bertino, I. Fovin, L. Provenza, Y. Saygin, and Y. Theodoridis, "State-of-the-Art in Privacy Preserving Data Mining," *Proc. ACM SIGMOD Record,* vol. 33, no. 1, pp. 50-57, Mar. 2004.

[22] C. Giannella, K. Liu, and H. Kargupta, "On the Privacy of Euclidean Distance Preserving Data Perturbation," *Computing Research Repository,* 2009.

[23] P. Samarati and L. Sweeney, "Protecting Privacy When Disclosing Information: K-Anonymity and Its Enforcement through Generalization and Suppression," 1998.

[24] H. Park and K. Shim, "Approximate Algorithms with Generalizing Attribute Values for K-Anonymity," *Information Systems,* vol. 35, no. 8, pp. 933-955, Dec. 2010.

[25] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "L-Diversity: Privacy Beyond K-Anonymity," *ACM Trans. Knowledge Discovery from Data,* vol. 1, Mar. 2007.

[26] J. Li and G. Ruhe, "Decision Support Analysis for Software Effort Estimation by Analogy," *Proc. Int'l Workshop Predictor Models in Software Eng.*, 2007.

[27] C. Dwork, "Differential Privacy," *Automata, Languages and Programming,* M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, eds., vol. 4052, pp. 1-12, Springer, 2006.

[28] C. Dwork, "Differential Privacy: A Survey of Results," *Proc. Fifth Int'l Conf. Theory and Applications of Models of Computation,* pp. 1-19, 2008.

[29] I. Dinur and K. Nissim, "Revealing Information While Preserving Privacy," *Proc. 22nd ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems,* pp. 202-210, 2003.

[30] A. Chin and A. Klinefelter, "Differential Privacy as a Response to the Reidentification Threat: The Facebook Advertiser Case Study," *North Carolina Law Rev.*, vol. 90, no. 5, 2012.

[31] F. Peters, "Cliff: Finding Prototypes for Nearest Neighbor Algorithms with Application to Forensic Trace Evidence," master's thesis, 2010, copyright ProQuest, UMI Dissertations Publishing, http://search.proquest.com/docview/859578571?accountid=2837, 2010.

[32] O. Jalali, T. Menzies, and M. Feather, "Optimizing Requirements Decisions with Keys," *Proc. Fourth Int'l Workshop Predictor Models in Software Eng.,* http://menzies.us/pdf/08keys.pdf, 2008.

[33] Y. Jiang, B. Cukic, and Y. Ma, "Techniques for Evaluating Fault Prediction Models," *Empirical Software Eng.,* vol. 13, pp. 561-595, 2008.

[34] T. Menzies, A. Dekhtyar, J. Distefano, and J. Greenwald, "Problems with Precision: A Response to "Comments on 'Data Mining Static Code Attributes to Learn Defect Predictors'," *IEEE Trans. Software Eng.*, vol. 33, no. 9, pp. 637-640, Sept. 2007.

[35] M. Jureczko and L. Madeyski, "Towards Identifying Software Project Clusters with Regard to Defect Prediction," *Proc. Sixth Int'l Conf. Predictive Models in Software Eng.*, pp. 9:1-9:10, 2010.

[36] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten, "The Weka Data Mining Software: An Update," *ACM SIGKDD Explorations Newsletter,* vol. 11, pp. 10-18, Nov. 2009.

[37] D. Lewis, "Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval," *Proc. 10th European Conf. Machine Learning,* pp. 4-15, 1998.

[38] J.C.J. Platt, "Fast Training of Support Vector Machines Using Sequential Minimal Optimization," *Advances in Kernel Methods,* B. Schölkopf, C.J.C. Burges, and A.J.A. Smola, eds., pp. 185-208, MIT Press, 1999.

[39] C. Bishop and G. Hinton, *Neural Networks for Pattern Recognition,* Clarendon Press, http://books.google.com/books?id=-aAwQO_-rXwC, 1995.

[40] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings," *IEEE Trans. Software Eng.,* vol. 34, no. 4, pp. 485-496, July/Aug. 2008.

[41] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A Systematic Literature Review of Fault Prediction Performance in Software Engineering," *IEEE Trans. Software Eng.,* vol. 38, no. 6, pp. 1276-1304, Nov.-Dec. 2012.

[42] E. Osuna, R. Freund, and F. Girosit, "Training Support Vector Machines: An Application to Face Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 130-136, June 1997.

[43] K. Taneja, M. Grechanik, R. Ghani, and T. Xie, "Testing Software in Age of Data Privacy: A Balancing Act," *Proc. 19th ACM SIGSOFT Symp. and 13th European Conf. Foundations of Software Eng.*, pp. 201-211, 2011.

[44] Q. Zhang, N. Koudas, D. Srivastava, and T. Yu, "Aggregate Query Answering on Anonymized Tables," *Proc. 23rd IEEE Int'l Conf. Data Eng.*, pp. 116-125, 2007.

[45] L. Sweeney, "Achieving K-Anonymity Privacy Protection Using Generalization and Suppression," *Int'l J. Uncertainty Fuzziness Knowledge-Based Systems*, vol. 10, no. 5, pp. 571-588, Oct. 2002.

[46] H. Brighton and C. Mellish, "Advances in Instance Selection for Instance-Based Learning Algorithms," *Data Mining and Knowledge Discovery*, vol. 6, pp. 153-172, 2002.

[47] J. Bezdek and L. Kuncheva, "Some Notes on Twenty One (21) Nearest Prototype Classifiers," *Proc. Joint IAPR Int'l Workshops Advances in Pattern Recognition*, pp. 1-16, 2000.

[48] J.C.J. Bezdek and L.I.L. Kuncheva, "Nearest Prototype Classifier Designs: An Experimental Study," *Int'l J. Intelligent Systems*, vol. 16, no. 12, pp. 1445-1473, http://dx.doi.org/10.1002/int.1068, 2001.

[49] S. Kim and B. Oommen, "A Brief Taxonomy and Ranking of Creative Prototype Reduction Schemes," *Pattern Analysis and Applications*, vol. 6, no. 3, pp. 232-244, Dec. 2003.

[50] J. Olvera-Lpez, J. Carrasco-Ochoa, and J. Martnez-Trinidad, "A New Fast Prototype Selection Method Based on Clustering," *Pattern Analysis and Applications*, vol. 13, pp. 131-141, 2010.

[51] H.B.H. Mann and D.R.D. Whitney, "On a Test of Whether One of Two Random Variables Is Stochastically Larger than the Other," *The Annals of Math. Statistics*, vol. 18, no. 1, pp. 50-60, http://www.jstor.org/stable/2236101, 1947.

[52] M. Castro, M. Costa, and J.-P. Martin, "Better Bug Reporting with Better Privacy," *Proc. 13th Int'l Conf. Architectural Support for Programming Languages and Operating Systems*, pp. 319-328, http://doi.acm.org/10.1145/1346281.1346322, 2008.

[53] J. Clause and A. Orso, "Camouflage: Automated Anonymization of Field Data," *Proc. 33rd Int'l Conf. Software Eng.*, p. 2130, 2011.

[54] A. Budi, D. Lo, L. Jiang, and Lucia, "Kb-Anonymity: A Model for Anonymized Behaviour-Preserving Test and Debugging Data," *Proc. 32nd ACM SIGPLAN Conf. Programming Language Design and Implementation*, pp. 447-457, http://doi.acm.org/10.1145/1993498.1993551, 2011.

[55] R. Agrawal and R. Srikant, "Privacy-Preserving Data Mining," *ACM SIGMOD Record*, vol. 29, no. 2., pp. 439-450, 2000.

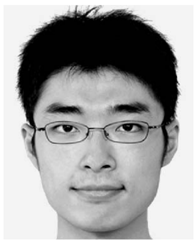
[56] S. Kim, H. Zhang, R. Wu, and L. Gong, "Dealing with Noise in Defect Prediction," *Proc. 33rd Int'l Conf. Software Eng.*, pp. 481-490, 2011.

[57] B. Boehm and P. Papaccio, "Understanding and Controlling Software Costs," *IEEE Trans. Software Eng.*, vol. 14, no. 10, pp. 1462-1477, Oct. 1988.

[58] J.B. Dabney, G. Barber, and D. Ohi, "Predicting Software Defect Function Point Ratios Using a Bayesian Belief Network," *Proc. Second Int'l Conf. Predictive Models in Software Eng.*, 2006.

Fayola Peters is working toward the PhD degree in computer science at West Virginia University. Her general areas of interest include empirical software engineering (ESE) and data mining. Specifically, she studies conclusion instability in ESE and finding solutions to privacy issues in software engineering. She is also one of the curators of the PROMISE repository (see http://promisedata.googlecode.com).

Tim Menzies received the PhD degree from the University of New South Wales, Sydney, Australia. He is a professor in computer science at West Virginia University (WVU) and the author of more than 200 referred publications. At the time of this writing, he is one of the 10 most cited authors in software engineering (out of 60,000+ researchers, see http://goo.gl/vggy1). At WVU, he has been a lead researcher on projects for NSF, NIJ, DoD, NASA, as well as doing joint research work with private companies. He teaches data mining and artificial intelligence and programming languages. He is the cofounder of the PROMISE conference series devoted to reproducible experiments in software engineering (see http://promisedata.googlecode.com). He is an associate editor of the *IEEE Transactions on Software Engineering*, the *Empirical Software Engineering* journal, and the *Automated Software Engineering* journal. In 2012, he served as a cochair of the program committee for the IEEE Automated Software Engineering conference. He is a member of the IEEE. For more information, see his web site http://menzies.us or his vita at http://goo.gl/8eNhY or his list of pubs at http://goo.gl/8KPKA.



Liang Gong is working toward the master's degree in the School of Software, Tsinghua University, Beijing, China. His current research interests include software fault localization and test case prioritization; he is also interested in software engineering and program analysis. More information can be found on his home page: http://gongliang3.appspot.com/.

Hongyu Zhang received the PhD degree from the National University of Singapore. He is an associate professor in the School of Software, Tsinghua University, Beijing, China. Before joining Tsinghua, he was a lecturer at RMIT University, Australia, and a research fellow at the National University of Singapore. His research interests include software engineering, in particular, software quality engineering, metrics, maintenance, and reuse. His research focuses on improving software quality and productivity. He has published approximately 60 research papers in international journals and conferences. He is a principle investigator for many national and university research projects. He is a member of the IEEE. More information about him can be found at his webpage at: https://sites.google.com/site/hongyujohn.