Privacy and Utility for Defect Prediction: Experiments with MORPH

Fayola Peters, Tim Menzies
Lane Department of Computer Science and Electrical Engineering,
West Virginia University, Morgantown, USA
fpeters@mix.wvu.edu, tim@menzies.us

***Abstract*—Ideally, we can learn lessons from software projects across multiple organizations. However, a major impediment to such knowledge sharing are the privacy concerns of software development organizations. This paper aims to provide defect data-set owners with an effective means of privatizing their data prior to release. We explore MORPH which understands how to maintain class boundaries in a data-set. MORPH is a data mutator that moves the data a random distance, taking care not to cross class boundaries. The value of training on this MORPHed data is tested via a *10-way within* learning study and a *cross* learning study using Random Forests, Naive Bayes, and Logistic Regression for ten object-oriented defect data-sets from the PROMISE data repository. Measured in terms of exposure of sensitive attributes, the MORPHed data was four times more private than the unMORPHed data. Also, in terms of the *f-measures*, there was little difference between the MORPHed and unMORPHed data (original data and data privatized by data-swapping) for both the *cross* and *within* study. We conclude that at least for the kinds of OO defect data studied in this project, data can be privatized without concerns for inference efficacy.**

***Keywords*-privacy; defect prediction; data mining**

I. Introduction

Within-company defect prediction is the means by which organizations predict the number of defects in their software. *Cross-company defect prediction* 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]–[3] (caveat: the cross-company data must be carefully filtered when being applied locally). The success of such cross-company learning experiments suggests that there exist general principles of software engineering that transcend project boundaries and which can be used to learn general models of software engineering.

However, before we conduct widespread cross-company learning experiments, we must first address the privacy concerns of data owners who are willing to share their data. Unless we can address these concerns, continued progress in this promising area will be stalled. Extracting project data from organizations is often very difficult due to the business sensitivity associated with the data. For example, for eleven years the second author worked at NASA's software IV&V center. NASA divides its IV&V work between multiple contractors who all work on similar projects. Every three years, all the contracts are subject to competitive bidding. Hence, those contractors are very reluctant to share data about (say) the effort associated with finding mission-critical errors in manned spacecraft, lest that data is used against them during the bidding process. Consequently, researchers and NASA managers suffer from a severe "data drought" that inhibits their ability to learn effective defect detection models for life-threatening software errors [4].

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 [5].

These examples are a clear indication of data owners who are willing to share their data for research purposes but are unable to do so because of privacy concerns. One way to convince data owners to publish their data is to offer them a means to privatize their data in such a way as to 1) prevent the competition from learning specific sensitive metric values from their released data and 2) ensuring that this privatized data remain useful for research purposes such as cross-company learning. With cross-company learning where the goal of a researcher is to find defects in a software system before deployment, all that is required of the data is that the relationship between the class values and attribute values remain intact. On the other hand, the attacker's goal is to seek out sensitive information without concerns for defect prediction.

We find we can exploit the difference between the goals of an attacker and the goals of defect prediction. We show that it is possible to secure against attackers while preserving the relationships required for effective defect prediction. MORPH [1] is a privacy algorithm designed to reduce the attacker's *trust* in the released data. In contrast, a researcher looking for defects in their software systems will still find the privatized data "useful". In the context of this paper, we will say that a data-set is "useful" if it can be used to learn defect predictors.

[1]The term MORPH in this work refers to data transformation via perturbation to create synthetic data.

This paper proposed MORPH as a sharing mechanism for privacy-sensitive data. Using MORPH, we can ensure privacy while supporting defect prediction since MORPH as an instance mutator perturbs all instance values by a random amount. This amount is selected to move an instance to a new location in n-space without moving it across the boundaries between classes. Potentially, MORPH increases the privacy of all mutated individuals since their original data is now distorted.

In support of the adoption of MORPH, we address three specific research questions:

- RQ1: *Does MORPH improve privacy?*
- RQ2: *Does MORPH improve privacy better than anything else?*
- RQ3: *Does MORPH degrade performance?*

To the best of our knowledge, this is the first report of a data mining method that increases the privacy of a data-set without damaging inference power. This is a significant result since prior studies [6] have reported that the application of standard privacy methods such as k-anonymity, l-diversity or t-closeness both damaged inference power, and offered little overall improvements in privacy. Hence, for learning from data that must be privatized, we recommend MORPHing over k-anonymity, l-diversity or t-closeness.

II. Background

In this section, we provide background on the following: a) defect prediction b) the cross company defect prediction process; c) privacy-preserving data publishing d) problems with privacy and provide e) attack models and f) utility models used in this paper.

A. Defect Prediction

Boehm & Papaccio advise that reworking software (e.g. to fix bugs) is cheaper earlier in the life cycle than later "*by factors of 50 to 200*" [7]. Other research makes the same conclusion. A panel at IEEE Metrics 2002 concluded that finding and fixing severe software problems after delivery is often 100 times more expensive than finding and fixing them during the requirements and design phase [8].

Defect prediction allows software companies to take advantage of early defect detection. Models made for defect prediction are built with *within-company* data-sets using common machine learners. The data-sets 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 data-set can then take a new unlabeled instance and label it as $defective$ or *not defective*. When evaluating the prediction models we use the 10*10 cross-validation technique. Here the defect data is divided into ten(10) folds. One fold is used as a test-set representing new unlabeled instances, while the other nine(9) folds are combined and act as the training-set used to create prediction models. The test instances are then labeled according to the models and these new labels can be compared to the original labels using various performance measures (discussed in Section II-F).

B. Cross Company Defect Prediction

When data can be shared between organizations, defect predictors from one organization can generalize to another. For example, defect predictors developed at NASA [9] 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% of the defects using just 40% of their QA effort (measured in staff hours) [10].

Such cross-company learning is useful since, as Zimmermann et. al. [11] observed, defect prediction via local data is not always available to many software companies as

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

Kitchenham et al. [12] also see problems with relying on *within-company* data-sets. They note that the time required to collect enough data on past projects from a single company may be prohibitive. Additionally, 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 cross-company learning were either very negative [11] or inconclusive [12]. Recently, we have had more success using better selection tools for training data [2], [3] but this success was only possible if the learner had unrestricted access to all the data. As discussed below, this is a problem.

C. Privacy Preserving Data Publishing

Data sharing across companies exposes the data provider to unwanted scrutiny. Some of these concerns reflect the low quality of our current anonymization technologies. For example, the state of Massachusetts once released some healthcare data, anonymized according to HIPPA regulations (http://goo.gl/HQiW6). When this supposedly "anonymized" data was joined to other data (voters lists and census data) it was possible to find data of prominent individuals such as former Massachusetts governor William Weld [13].

We say that *re-identification* occurs when an attacker with external information such as census data can identify an individual from a privatized data-set. For the kinds of data used in this study (which are aggregated at the project level, but not at the developer level), these data-sets do not contain personnel information. Hence, re-identification of individuals is not explored further in this study.

On the other hand, *sensitive attribute value disclosure* is of great concern with the data used in this study. This is where an individual in a data-set can be associated with a sensitive attribute value; e.g. to identify the quality attributes of corporations in data-sets. Such sensitive attribute value disclosure can prove problematic. Some of the metrics contained in defect data can be considered as *sensitive* to the data owners. These can include any size measures such as lines of code (loc) or cyclomatic complexity (max-cc or avg-cc). If these size measures are joined to development time, it would be possible for rivals competing on a bid to discredit their opposition by revealing (say) slower development times amongst their competitors.

The goal of privacy preserving data publishing (PPDP) is to ensure that *the published data can preserve individual privacy while remaining useful for workloads such as data mining [13]–[15]* or defect prediction. The data is looked at as having three components: 1) quasi identifiers (QIDs) such as age, and zip code that help identify an individual in a data-set; 2) sensitive attributes (S), that are a privacy threat when matched to an individual (e.g. medical diagnosis); 3) and non-sensitive attributes. As discussed next, the goals of PPDP are hard to reach.

D. Problems with Privacy

Many researchers comment on how privatization algorithms can distort data. For example, consider privatization via these three methods: 1) Replacing exact numeric values with intervals that cover a range of values; e.g. 17 might become *15-20*; 2) Replacing symbols with more general terms; e.g. "date of birth" becomes "month of birth"; or "engineer" or "lawyer" becomes "professional"; 3) Suppressing exact values by replacing them; e.g. replace specific values with "don't know" [16]; or perturbing specific values by an amount selected randomly from a distribution.

According to [13], these methods hide potentially important details in the QID that can confuse a data miner. Worse, these transforms may not guarantee privacy. For example, consider privacy-via-perturbation (e.g. data-swapping). 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 [17].

Widely-used privatization approaches include *k-anonymity*, *l-diversity* and *t-closeness*. K-anonymity [18] makes each record in the table be indistinguishable with *k-1* other records by suppression or generalization [18]–[20]. The limitations of k-anonymity, as listed in [6] are many fold and include the fact that it does not hide whether a given individual is in the database. Also, in theory, k-anonymity hides uniqueness (and hence identity) in a data-set, 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 [6].

Machanavajjhala et al. [21] 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 correct sensitive attribute value of any member of a QID group.

Work by Li et al. [22], later showed that *l-diversity* was vulnerable to *skewness* and *similarity* attacks making it insufficient to prevent attribute disclosure. Hence, Li et al. 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 are 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 and so causes degradation of data utility [22].

In practice, the above issues with privacy algorithms are very real problems. Grechanik et al. [23] found that k-anonymity greatly degraded the test-coverage of data-centric applications. Furthermore, Brickell and Shmatikov [6] report experiments where to achieve 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 vs. trivial privatization. Worse, they also reported that simplistic trivial privatization provides better privacy results than supposedly better methods like l-diversity, t-closeness and k-anonymity.

The reason for this poor performance of such widely explored privacy algorithms is unclear. We speculate that the empirical basis for certifying these standard privacy methods may not be very strong. Fung et al. [13] report that one data-set (the 48,842 records of ADULT; see http://goo.gl/1XZT7) is the "de facto benchmark for testing anonymization algorithms" and list 13 papers that use it as the only test case for their algorithms. In this regard, our results have more external validity since we base our experiments on ten different data-sets (caveat: they have some similarities in that they are all are open source object-oriented JAVA projects).

E. Attack Models

To precisely define privacy, we must first define how we model an attacker trying to access data. In this section we review three standard attack models from the literature [6], [21], [22] and present the one used in this work which was adapted from Brickell and Shmatikov [6] (we will use this last one since it includes and extends the others).

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

Before continuing, note that examples used to explain the following attack models and later on sensitive attribute disclosure (Section **??**), are based on the Top and $Bottom$ tables of Figure 2. The data-set used is an abbreviated $ant13$ defect data-set from the PROMISE data repository (http://goo.gl/mStAY). In the examples to follow Top represents an original data-set while $Bottom$ is the 2-anonymity version of Top.

1) Machanavajjhala et al.'s Attack Models: The homogeneity attack and the background knowledge attack are defined in [21] and are seen as an attack on k-anonymity. The homogeneity attack results in groups that leak information due to the lack of diversity in the sensitive attribute. For an example of this kind of attack, consider Figure 2. If an attacker knows that a target file has the value of 11 for rfc and zero for $lcom$ ($Q(2) = \{rfc^{11}, lcom^{0}\}$), the $Bottom$ table produces two possibilities for the target file (types.EnumeratedAttribute and NoBannerLogger). Since both have the same sensitive attribute value the attacker is 100% sure that the target file has 59 lines of code.

The background knowledge attack is explained by the following example. Using the 2-anonymity data of Figure 2, suppose the attacker has the following knowledge about a target file: $Q(1) = \{cbo^{8}\}$. Here Q(1) says that the query size is 1 and *cbo* is a QID with the value 8. From the results of that query, the attacker knows that the target is contained in the first group of files (taskdefs.ExecuteOn, taskdefs.Cvs and DefaultLogger) with sensitive values of 395, 310 and 257 respectively. Assuming that the attacker has additional knowledge that files with *cbo* values ≤ 8 tend to also have *loc* less than 300, then the attacker can be almost certain that the target file has 257 lines of code.

2) Li et al.: Similarity Attack: Unlike the previous attack models the similarity attack explored by Li et al. [22] is considered as an attack on l-diversity. This attack occurs when sensitive attribute values in a QID group are distinct but semantically similar, for instance, if the sensitive attribute values fall within a narrow range. Consider the following example: Suppose the 2-anonymity data in Figure 2 consisted of all but the last two records. This will leave a table that conformed to 2-diversity. If an attacker poses the query: $Q(1) = \{cbo^{8}\}$, the first group of files is returned as before. With this result, an attacker will know with 33% guarantee that the target has either 395, 310 or 257 lines of code. Although this may seem like a good result in terms of privacy, since *loc* is a numerical attribute, this narrow range of results can be considered as sensitive information being revealed [24]. This is because the attacker can infer that the target file has a relatively high number for *loc*.

3) Brickell and Shmatikov Attack: This attack model is based on the previous models. 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 data set and also supplied with the private data-set.

ID	Quasi Identifiers (QIDs)								S
name	wmc	dit	noc	cbo	rfc	lcom	ca	ce	loc
taskdefs. ExecuteOn	11	4	2	14	42	29	2	12	395
Default Logger	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. Enu-merated Attribute	5	1	5	12	11	8	11	1	59
NoBanner Logger	4	2	0	3	16	0	0	3	59

ID	Quasi Identifiers (QIDs)								S
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 Logger	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. Enu-merated Attribute	<7	<5	≤5	*	11-16	≤8	*	≤4	59
NoBanner Logger	<7	<5	≤5	*	11-16	≤8	*	≤4	59

Figure 1: Privatization example using an abbreviated $ant13$ data-set from the PROMISE data repository (http://goo.gl/mStAY). The number of lines of code (loc) is the sensitive attribute (S). **Top**: This represents the original data-set. **Bottom**: This is the original data-set after 2-anonymity using generalization and suppression.

In order to keep the privatized data-set $truthful$, 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 MORPH, we apply Equal Frequency Binning (EFB)[3] to the sensitive attribute to create ten(10) sub-ranges of values in order to easily report on the privacy level of the privatized data-set.

F. Utility Model

Utility is the measure of the "usefulness" of a data-set after privatization. An inherent understanding in the field of PPDP is that there exists a trade-off between privacy and utility in that with increased privacy, the utility of a data-set decreases [6]. Therefore a measure of privacy alone is not enough to judge the success of a privacy algorithm: we also need to judge the *utility* of the result.

Like Brickell and Shamtikoc, we measure the *utility* of our privatized data-set empirically. Using Random Forests, Naive Bayes and Logistic Regression, we create defect

[3]A column of data is divided at the 10,20,30,...,90-th percentile to create ten bins with equal frequency.

prediction models for the privatized and original data-sets. These learners were chosen since we wanted to test the impact of our privacy tools on a wide range of learners including *iterative dichotomizers* such as Random Forests; *statistical learners* such as Naive Bayes; Neighbors; and *parametric methods* such as logistic regression.

We then compare the performance of our defect predictors using *f-measures*. An f-measure is the harmonic mean of $precision$ and $recall$. These three measures are defined and calculated as follows:

- Let A, B, C and D represent true negatives, false negatives, false positives and true positives respectfully;
- $Recall$, which reveals how much of the target was found, is the result of true positives divided by the sum of false negatives and true positives, *D / (B + D)*;
- $Precision$ reveals how many of the rows that triggered the detector actually contained the target concept. It is the result of true positives divided by the sum of false positives and true positives, *D/ (C + D)*.
- *f-measure* allows for a dual assessment of both $recall$ and $precision$ ($\frac{2 \times precision \times recall}{precision + recall}$). It has the property that if either precision or recall is low, then the f-measure is decreased.

Note that in those comparisons, we set aside some test data, privatize the remaining training data, then perform our tests on the privatized data.

III. MORPH Design and Privacy Framework

This section introduces MORPH, a privacy algorithm for *defect data*. It is based on a Nearest Unlike Neighbor (NUN) [25] approach (explained in Section III-A) coupled with randomization. MORPH has two goals: achieve a proficient level of privacy while maintaining the utility of the defect data. In other words, defect predictors built from the privatized data must be as good as those from the original data-set. Further, the privatized data should reveal little or no excess information than the original data. To fulfill these requirements, MORPH changes each row of the original data just enough to avoid a change in its outcome, that is, its class (defects) label must not change.

In Section III-A, we describe MORPH in detail. This is followed by the privacy metric used to measure its performance.

A. Privacy Algorithm

MORPH acts to preserve the general topology of the space of the instances in the data, while changing the location of specific instances within that space. MORPH is an instance mutator that changes the attribute values of each instance by replacing these original values to $MORPHed$ values. MORPH takes care never to change an instance such that it crosses the boundary between the original instance and instances of another class.

The MORPHed instances are created by applying Equation 1 to each attribute value of the instance.

$$y_i = x_i \pm (x_i - z_i) * r \tag{1}$$

Let $x \in D$ be the original instance 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 (and distance is calculated using the $Euclidean$ distance). The random number r is calculated with the property $\alpha \leq r \leq \beta \leq \textit{delta}/2$, where $\alpha = 0.15$ and $\beta = 0.35$.

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 privatized data-set.

B. Privacy Measure

In this work we measure privacy in terms of *sensitive attribute disclosure* (SAD). Here we determine if an attacker's query will allow them to identify a group of individual files in a data-set and associate them with a sensitive attribute value sub-range. In the following sections we describe how the attacker's knowledge is modeled using random queries to measure SAD.

1) The Query Generator: 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 QIDs in the data-set. As a result these attribute values are unchanged in the privatized data-set.

For the purposes of experimentation, we will assume that each query can be of length 1, 2 or 4 (denoted $Q(1)$, $Q(2)$ and $Q(4)$ respectively) and are made up of randomly selected attribute value pairs. To represent the attacker's knowledge, we first discretize the original data set using EFB. For these experiments, we will create 10 equal frequency bins.

After binning is complete, queries of lengths 1, 2 and 4 are randomly generated. For each query length we generate up to 1000 queries because it is not practical to test every possible query (with these data-sets the number of possible queries with arity 4 and no repeats is $42,296,805$).

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

- They must not include attribute value pairs from neither the designated sensitive attribute nor the class attribute;
- They must return at least 2 instances after a search of the original data-set;
- They must not be the same as another query no matter the order of the individual attribute value pairs in the query.

2) Sensitive Attribute Disclosure: Once a query is generated, it is used on the original and privatized data-sets to select instances that adhere to the query. SAD is then measured for each of the ten(10) sensitive attribute sub-ranges using the following formula:

$$SAD_i = \frac{\#v_i}{\#G} \tag{2}$$

where $\#V_i$ is the number of rows with a specific sensitive attribute sub-range, and $\#G$ is the number of rows selected by a query.

The intuition here is that an attacker will use a "best guess" approach i.e. find the sensitive attribute sub-range with the highest SAD using Equation 2. If a query results in the same sub-range being selected for both the original and privatized data-sets then we consider this to be a breach of privacy. On the other hand, if they are different, then the attack is unsuccessful and privacy is preserved.

Here is an example for one query: Using the tables in Figure 2, we create sensitive attribute sub-ranges $[58-136]_0$ and $[257-395]_1$. Assume that an attacker has the following query from our original data-set, Top: $Q(1) = \{cbo^{8-12}\}$. Applying this query to Top produces the following SAD results: $SAD_0 = \frac{1}{3}$ and $SAD_1 = \frac{2}{3}$. Here, the highest SAD returns the range [257-395]. Applying this query to the privatized data-set, $Bottom$, produces the following SAD results: $SAD_0 = \frac{1}{1}$ and $SAD_1 = \frac{0}{1}$. Here, the highest SAD returns the range [58-136]. Since the ranges returned are different we say that the attacker's query did not breach the privacy of the data-set and so the attacker's "best-guess" was wrong.

3) Increase Privacy Ratio: Finally, we can define the IPR. When all the sensitive attribute sub-ranges for each query are selected via the attacker's "best-guess" strategy, they are compared. If the query returns the same sub-range for the original and the privatized data-sets this is counted as a privacy breach. We then find the total number of breaches from all the queries and return the percentage.

Next, for the purposes of reporting the effects of MORPH on privacy, we say that the IPR is the ratio of the baseline of full disclosure i.e. the ability of the attacker to guess correctly for all queries; i.e. 100% to the percent of *actual correct guesses* by the attacker:

$$IPR = \frac{100}{Correct} \tag{3}$$

Based on our discussions with business users on different methods for defining privacy, we prefer Equation 3 to, say, the entropy measures used by Clause et al. [26]. Our users find the simplicity of the above expression easier to understand than Shannon's entropy.

IV. Experimental Evaluation

In this section we evaluate MORPH using a *within* and *cross* data study. To show how our privacy framework performs against a state-of-the-art privacy algorithm, we compare our work to *data-swapping*, a standard perturbation technique used for privacy [13], [24], [27] (discussed in Section IV-A). In our experiments, the WEKA [28] implementation of Random Forests, Logistic Regression and Naive Bayes are used to create defect models. In order to assist replication, our data comes from the on-line PROMISE data repository (http://goo.gl/mStAY). Figure 3 lists those data-sets and Figure 4 describes their attributes. The class labels are made categorical: "0" denotes no defects and "1" denotes more than zero defects.

Using these learners, we conducted two studies:

- For the *within*, a 10 × 10 cross-validation experiment was conducted where the defect data was shuffled and then separated into ten bins. For each bin, MORPH is applied to nine of the ten bins and tested on the remaining bin. In order to avoid *order effects* (where the conclusions are a result of some serendipitous ordering of the data), this process is repeated 10 times using different random shuffles of the rows in the data-sets.
- For the *cross* study, for all data-sets, each acts as a test-set while the remaining data-sets are training sets used as cross company data.

Recall from the introduction that our results are geared toward answering the following research questions:

- RQ1: Does MORPH improve privacy?
- RQ2: Does MORPH improve privacy better than anything else?
- RQ3: Does MORPH degrade performance?

RQ1 and R2 accounts for the privacy level of the privatized defect data-sets. This is measured via an attack model to determine if an attacker can associate an instance in the privatized data-set to a sensitive attribute value range. With RQ1 and R2 we directly address the main concern of data owners, "Will privacy be preserved after data has been published?"

Unlike RQ1 and R2, in RQ3 we examine the performance of defect models created from privatized data-sets based of the original data. The goal here is to show data owners and researchers, that privatized data can be used to make

Data	Symbol	Instances	Attributes	Class	Defect%
ant13	ant	125	20	2	20.2
arc	arc	234	20	2	11.5
camel10	cam	339	20	2	4
poi15	poi	237	20	2	40.5
redaktor	red	176	20	2	15.3
skarbonka	ska	45	20	2	20
tomcat	tom	858	20	2	9
velocity14	vel	196	20	2	25
xalan24	xal	723	20	2	15.2
xerces12	xer	440	20	2	16.1

Figure 2: Defect Data Set Characteristics

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

Figure 3: The C-K metrics of the data-sets used in this work (see Figure 3). The last row is the dependent variable.

adequate defect models which produce comparable and/or negligibly degraded results to the original data.

A. Data Swapping

In order to benchmark our approach, we need to compare it against some alternative. Since Grenchanik et al. and Brickell and Shmatikov [6], [23] report that k-anonymity, l-diversity, and t-closeness damage the utility of a data-set, we elect to use another privacy algorithm.

Since MORPH is a perturbation technique for generating synthetic data, for our comparisons, we implemented *data swapping*, a standard perturbation technique used for privacy [13], [24], [27]. This is a permutation approach that de-associates the relationship between a 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%.

B. Defect Predictors

Many machine learning techniques have been used for the purpose of defect prediction [29]. For this work we focus on Random Forest, an ensemble method, and two statistical methods, Naive Bayes and Logistic Regression. Recall, these were chosen since they are very different algorithms (iterative dichotomization, statistical, parametric). The WEKA implementation of these methods are used along with any default values [30]. Below is a brief description of these methods. Detailed descriptions can be found elsewhere [31].

- **Random Forests (RF)**: Breiman [31] describes RF as a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. Each tree in the collection is used to classify a new instance. The forest then selects a classification by choosing the majority result.
- **Naive Bayes (NB)**: Lewis [32] 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.
- **Logistic Regression (LR)**: Afzal [33] describes LR as a method to be used when the dependent variable is dichotomous (e.g. either fault-prone or non-fault-prone). The method avoids the Gaussian assumption used in standard Naive Bayes.

C. Methodology

In the field of Privacy-Preserving Data Publishing, it is often remarked that there is a tension between gain in privacy results and degradation in the utility of that data [6]. Therefore one cannot report on the privacy of a data-set without considering its effects on data utility i.e. the "usefulness" of the privatized data. To that end, we report two sets of results:

- The *increased privacy ratio*, measured via Equation 3;
- The *utility* of the learning, measured via the *f-measure*.

Algorithms	Symbol	Meaning
MORPH	m	data privatized by MORPH
swapped$X$	sX: s10, s20, s40	X represents the percentage of the original data swapped

Figure 4: Algorithm Characteristics

Note that prior results reported that attempts to increase privacy tended to damage utility [6].

D. Results

This section presents the results of our experiments. Before going forward, Figure 5 shows the notation and meaning of the algorithms used in this work.

RQ1: Does MORPH improve privacy? Figure 8 shows the increase privacy ratios seen in these experiments. To generate that figure we first privatize our data-sets using MORPH and data-swapping guided by the policy that the sensitive attribute values of loc and the $defects$ attribute remain unchanged after privatization. Next, using the query generator described in Section II, queries (the attackers' background knowledge) are drawn from the original data-sets. Applying Equation 3 we then calculated the increased privacy ratio by comparing the original and privatized data sets. The sub-ranges with the highest SADs from each are chosen and compared using Equation 3 (so our results are in that sense worst case since we focus on the largest SADs and the least increased privacy ratios).

The results of those calculations are shown in Figure 8. Interestingly, the *more knowledgeable* the attacker, the *less they can learn* from MORPHed data or data privatized by data-swapping. As we increase the background knowledge of the attacker (from one attribute to two to four), the increased privacy ration *increases*. This is due to the randomization of the data provided by MORPH. As we push data into random corners, it becomes less lucky that larger and larger conjunctions will find items of interest in the MORPHed data. The same can be said for the data-swapping algorithms.

From these results, the worse-case scenario for privacy is the attacker with the most general knowledge (i.e. fewest constraints, smallest queries). In that worse case, the size of the query is one and the median increased privacy ratio for MORPH is 4.4 while the data-swapping algorithms(s10, s20 and s40) are 2.8, 3.3 and 4.2 respectively. That is, even in the worst case MORPH is able to a make data-set at least four times more private than the original data-set.

RQ2: Does MORPH improve privacy better than anything else? It is important to note that RQ2 says *anything else* and not *everything else*. While we have shown that MORPH performs better than data swapping for s10 and s20, and has comparable results for s40 (see Figure 8), we have not explored the space of all possible data privacy algorithms. This paper has compared to privacy methods that have some successful track record (i.e. we compared to data swapping but not, say, k-anonymity since the Brickell and Shamtikoc [6] results were so negative on that approach). In future work, we will explore a broader range of methods.

RQ3: Does MORPH degrade performance? Figure 6 and Figure 7 report high *f-measures* for both cross and within learning with two exceptions:

1) the *within* Naive Bayes experiment in Figure 6b with *f-measures* below 55% for the $poi15(poi)$ data-set;
2) the *f-measures* for $poi15$ and $velocity14(vel)$ for the $cross$ experiment which uses Random Forests for defect prediction (see Figure 7).

Also shown on those figures are the *f-measures* seen in the privatized data. Both MORPH and the data-swapping algorithms perform just as well as the original data-sets in most cases for both the $within$ and $cross$ experiments. With MORPH however, poi15 is somewhat different to the other data-sets since, when it was used as a training set for cross-company learning with $redaktor(red)$ and $skarbonka(ska)$ (see Figure 7), it performed worse than all the other results. The reason for this outlier is unknown and will be subject of our future work. Otherwise, for nine out of ten training sets, those privatized by MORPH remain as useful as the original data.

V. 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:

First, sampling bias threats any data mining experiment; i.e., what matters there may not be true here. For example, the data-sets used here comes from the PROMISE repository and 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 in order to repeat, refute, or improve our results.

Second, another source of bias in this study is the learner used for the defect prediction studies. Data mining is a large and active field and any single study can only use a small subset of the known data mining algorithms. In this work, results for Naive Bayes, Random Forests and Logistical Regression are published.

Third, the field of privacy-preserving data publishing is active, and so it would be difficult compare the performance of MORPH to all of them. In this work we chose to use data-swapping as a baseline to judge the performance of our method.

Last, the utility of a privatized data-set can be measured semantically (where the workload is unknown) or empirically (known workload e.g. classification or aggregate query answering). In this work we measure utility empirically for defect prediction. As a result we cannot guarantee positive outcomes for other utility measures.

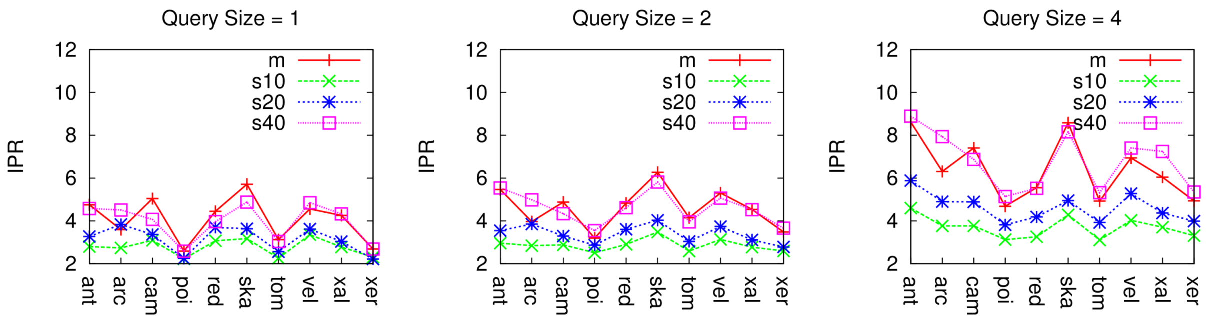

Figure 5: Increased Privacy Ratio (ipr) of Query sizes 1, 2 and 4.

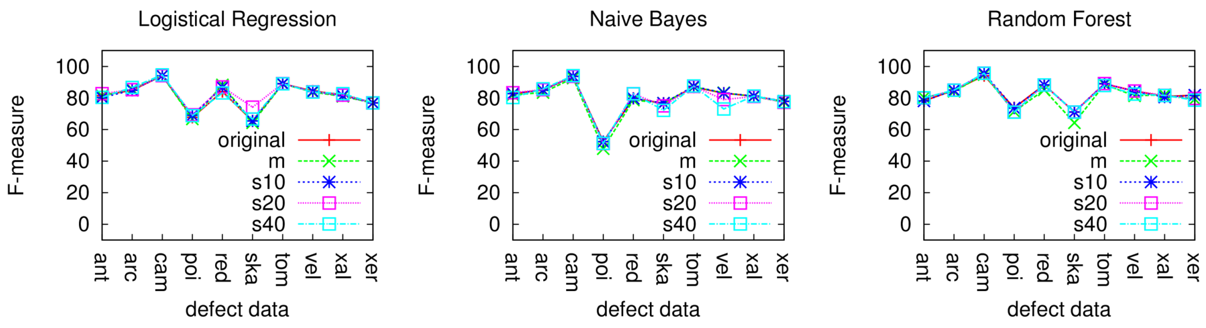

Figure 6: *Within* results: F-measures compared among the original and privatized data seen in 10*10-way cross-validation experiments.

VI. Related Work

A. Privacy Research in Software Engineering

To the best of our knowledge this is the first paper to address the issue of privacy in cross company defect prediction. However it is closely related to privacy work for software testing and debugging [26], [27]. Here although the work uses *within company* data, the privacy becomes an issue when it involves outsourcing the testing to third parties as is the case with [27] or collecting user information after a software system has been deployed [26]. In the former case, since companies do not wish to release actual cases for testing, they anonymize the test cases before releasing them 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 cross company prediction can suffer the same fate.

Clause et al. [26] presented an algorithm which anonymizes input sent from users to developers for debugging. Since the information from the user is likely to contain sensitive information, Clause et al. proposed a method which relies on the premise that there is more than one path to a “bug”. In other words their aim is 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. Next, the author applied a technique called *breakable input conditions* to ensure that the constraints do not select values from the original input data.

In contrast to the work done in [26], Taneja et. al. proposed PRIEST. 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 data-set 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 to our work 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 remains after privatization.

MORPH takes a different approach to the above research. Firstly, the techniques of Clause et al. assumes detailed connection knowledge between parts of a system. Such detailed connection knowledge is not present in defect data-sets like Figure 4. As to PRIEST, we have some reservations to data-swapping methods that use fixed distributions since Zhang & Zhao argue that that distribution can be reverse engineered from the privatized data [34].

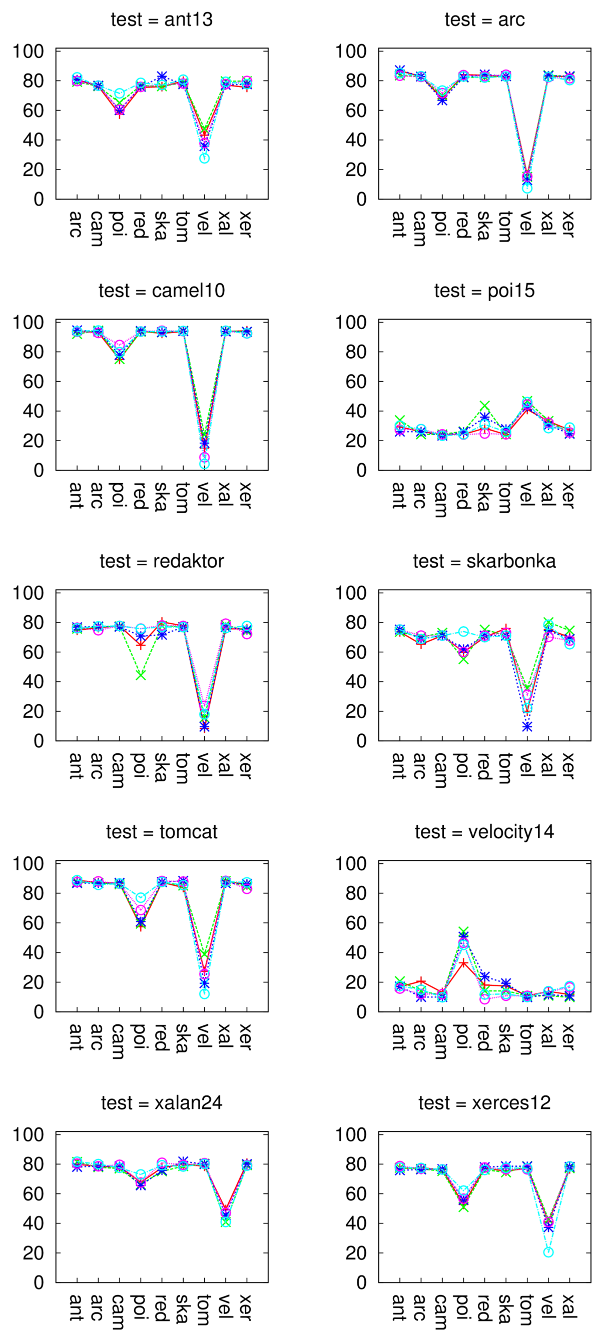

Figure 7: *Cross* results: The y-axis shows F-measures compared among the original and privatized data. Here Random Forests is chosen to build the defect models. Naive Bayes and Logistic Regression have similar results but these are left out due to the space constraint. KEY: original —+— , m --×-- , s10 ···*··· , s20 ···○··· , s40 ···○···

VII. Conclusion

Studies have shown that early detection and fixing of defects in software projects is less expensive than finding defects later on [7], [8]. Organizations with local data can take full advantage of this early detection benefit by doing within-company defect prediction. When an organization does not have enough local data to build defect predictors, they might try to access relevant data from other organizations in order to perform cross-company defect prediction. 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 data is privatized the less useful the data becomes.

To address this issue, in this paper we presented MORPH, a privacy algorithm designed to privatize defect data-sets for cross company defect prediction. Unlike previous studies, we show that MORPH (1) increases data privacy of data without (2) damaging our ability to reason about that data. Note that this is a significant result since prior work with the standard privatization technologies could not achieve those two goals.

Our work was framed in terms of three research questions:

- RQ1: *Does MORPH improve privacy?* Measured in terms of exposure of sensitive attributes, it was shown in Figure 8 that MORPHed data is four times (or more) private than the original data.
- RQ2: *Does MORPH improve privacy better than anything else?* Figure 8 shows that MORPH does better than s10 and s20. However it is comparable to s40.
- RQ3: *Does MORPH degrade performance?* We showed in Figure 6 and Figure 7 that for the purposes of learning defect predictors, training on the MORPHed data is just as effective as training on the unMORPHED data; i.e. MORPHing does *not* degrade inference efficacy.

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

Our results suggest the following future work:

- The experiments of this paper should be repeated on other data-sets.
- The current NUN algorithm is $O(N^2)$. We are exploring ways to optimize that with some clustering index method (e.g. k-means).
- While Figure 8 showed that we can increase privacy, they also showed that we cannot 100% guarantee it. At this time, we do not know the exact levels of privacy required in industry or if the results of Figure 8 meet those needs. This requires further investigation.

Acknowledgments

The work was funded by NSF grant CCF:1017330 and the Qatar/West Virginia University research grant NPRP 09-12-5-2-470.

References

[1] E. Kocaguneli and T. Menzies, "How to find relevant data for effort estimation?" in *Proceedings ESEM'11 (to appear)*, 2011.

[2] B. Turhan, T. Menzies, A. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empirical Software Engineering*, vol. 14, pp. 540–578, 2009.

[3] E. Kocaguneli, G. Gay, T. Menzies, Y. Yang, and J. W. Keung, "When to use data from other projects for effort estimation," in *Proceedings of the IEEE/ACM international conference on Automated software engineering*, ser. ASE '10. New York, NY, USA: ACM, 2010, pp. 321–324.

[4] T. Menzies, M. Benson, K. Costello, C. Moats, M. Northey, and J. Richarson, "Learning better IV&V practices," *Innovations in Systems and Software Engineering*, March 2008.

[5] 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 Engineering*, October 2008.

[6] J. Brickell and V. Shmatikov, "The cost of privacy: destruction of data-mining utility in anonymized data publishing," in *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '08. New York, NY, USA: ACM, 2008, pp. 70–78.

[7] B. Boehm and P. Papaccio, "Understanding and controlling software costs," *IEEE Trans. on Software Engineering*, vol. 14, no. 10, pp. 1462–1477, October 1988.

[8] F. Shull, V. B. ad B. Boehm, A. Brown, P. Costa, M. Lindvall, D. Port, I. Rus, R. Tesoriero, and M. Zelkowitz, "What we have learned about fighting defects," in *Proceedings of 8th International Software Metrics Symposium, Ottawa, Canada*, 2002, pp. 249–258.

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

[10] A. Tosun, B. Turhan, and A. Bener, "Practical considerations in deploying ai for defect prediction: a case study within the turkish telecommunication industry," in *Proceedings of the 5th International Conference on Predictor Models in Software Engineering*, ser. PROMISE '09. New York, NY, USA: ACM, 2009, pp. 11:1–11:9.

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

[12] B. A. Kitchenham, E. Mendes, and G. H. Travassos, "Cross versus within-company cost estimation studies: A systematic review," *IEEE Transactions on Software Engineering*, vol. 33, pp. 316–329, 2007.

[13] 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.

[14] J. Domingo-Ferrer and U. Gonzalez-Nicolas, "Hybrid microdata using microaggregation," *Information Sciences*, vol. 180, no. 15, pp. 2834–2844, 2010.

[15] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan, "Workload-aware anonymization techniques for large-scale datasets," *ACM Transactions on Database Systems*, vol. 33, no. 3, pp. 1–47, 2008.

[16] V. Verykios, E. Bertino, I. Fovin, L. Provenza, Y. Saygin, and Y. Theodoridis, "State-of-the-art in privacy preserving data mining," *SIGMOD RECORD*, vol. 33, no. 1, pp. 50–57, MAR 2004.

[17] C. Aggarwal and P. Yu, "A general survey of privacy-preserving data mining models and algorithms," in *Privacy preserving data mining: models and algorithms*. Springer-Verlag, 2008, pp. 11–43.

[18] L. Sweeney, "k-anonymity: A model for protecting privacy," *Ieee Security And Privacy*, vol. 10, no. 5, pp. 557–570, 2002.

[19] P. Samarati and L. Sweeney, "Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression," 1998.

[20] 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.

[21] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "L-diversity: Privacy beyond k-anonymity," *ACM Trans. Knowl. Discov. Data*, vol. 1, March 2007.

[22] N. Li and T. Li, "t-closeness: Privacy beyond k-anonymity and -diversity," in *In Proc. of IEEE 23rd Intl Conf. on Data Engineering (ICDE07*, 2007.

[23] M. Grechanik, C. Csallner, C. Fu, and Q. Xie, "Is data privacy always good for software testing?" in *Proceedings of the 2010 IEEE 21st International Symposium on Software Reliability Engineering*, ser. ISSRE '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 368–377.

[24] Q. Zhang, N. Koudas, D. Srivastava, and T. Yu, "Aggregate Query Answering on Anonymized Tables," *2007 IEEE 23rd International Conference on Data Engineering*, pp. 116–125, 2007.

[25] B. V. Dasarathy, *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, 1991.

[26] J. Clause and A. Orso, "Camouflage : Automated anonymization of field data," *Proceeding of the 33rd international conference on Software engineering*, p. 2130, 2011.

[27] K. Taneja, M. Grechanik, R. Ghani, and T. Xie, "Testing Software In Age Of Data Privacy: A Balancing Act," *Public Policy*, 2011.

[28] I. H. Witten and E. Frank, *Data mining. 2nd edition*. Los Altos, US: Morgan Kaufmann, 2005.

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

[30] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, pp. 10–18, November 2009.

[31] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[32] D. Lewis, "Naive (bayes) at forty: The independence assumption in information retrieval," in *Machine Learning: ECML-98*, ser. Lecture Notes in Computer Science, C. Ndellec and C. Rouveirol, Eds. Springer Berlin / Heidelberg, 1998, vol. 1398, pp. 4–15.

[33] W. Afzal, "Using faults-slip-through metric as a predictor of fault-proneness," in *Proceedings of the 2010 Asia Pacific Software Engineering Conference*, ser. APSEC '10, 2010, pp. 414–422.

[34] N. Zhang and W. Zhao, "Privacy-preserving data mining systems," *Computer*, vol. 40, no. 4, pp. 52–58, April 2007.