Case Consistency: A Necessary Data Quality Property for Software Engineering Data Sets

Passakorn Phannachitta† Akito Monden† Jacky Keung‡ Kenichi Matsumoto†

†Graduate School of Information Science
Nara Institute of Science and Technology
Japan
{phannachitta-p, akito-m, matumoto}
@is.naist.jp

‡ Department of Computer Science
City University of Hong Kong
Hong Kong SAR
Jacky.Keung@cityu.edu.hk

ABSTRACT

Data quality is an essential aspect in any empirical study, because the validity of models and/or analysis results derived from an empirical data is inherently influenced by its quality. In this empirical study, we focus on data consistency as a critical factor influencing the accuracy of prediction models in software engineering. We propose a software metric called Cases Inconsistency Level (*CIL*) for analyzing conflicts within software engineering data sets by leveraging probability statistics on project cases and counting the number of conflicting pairs. The result demonstrated that *CIL* is able to be used as a metric to identify either consistent data sets or inconsistent data sets, which are valuable for building robust prediction models. In addition to measuring the level of consistency, *CIL* is proved to be applicable to predict whether or not an effort model built from data set can achieve higher accuracy, an important indicator for empirical experiments in software engineering.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*Performance measures*

General Terms

Management, Measurement, Experimentation

Keywords

Data quality; Data consistency; Software effort estimation; Heterogeneous distance function; Empirical software engineering

1. INTRODUCTION

In quantitative software engineering methodologies, accumulated historical data from previous software projects are an important information source supporting new software development. As opposed to a qualitative way that commonly uses human factors such as expert opinions [16], the quantitative approach being focused in this study continually reused data extracted from software development processes in the past. The use of this information helps project managers to gain a better understanding of a new project. However, more important than the amount of data being available for analyzing is the quality of the data itself. According to Gartner [38], 40% of business initiatives could never reach their desirable target due to poor data quality, which is often caused by issues such as data inconsistencies, lack of completeness, and redundant records.

Evaluating the data set quality is an important process before applying analyses or estimation to the data. According to a taxonomy of data quality challenges in empirical software engineering (ESE) proposed by Bosu and MacDonell [6], data quality issues in the existing research are generally considered outliers [25], noise [19, 26], data incompleteness [26, 36], data inconsistency [8, 41], and redundant data [12]. Numerous attempts to handle outliers, noise, data incompleteness, and redundant data have been proposed in ESE literatures. However, data set inconsistency, one of the most essential problems that exist in data sets [7, 8, 37, 41] does not have a clear solution for its handling.

In this work, we introduce Cases Inconsistency Level (*CIL*), a software metric based on an analysis between all the possible paring of the project cases to quantify the level of inconsistency of ESE project data. Inconsistent data in this study is defined as the data consisting of any two project cases which conflicting between their attribute values. We use software effort estimation data sets to demonstrate the concept of our proposed metric. Software effort estimation is a task to construct a predictive model using historical quantitive data, and the model is then used for estimating an effort value for a new software project. In the case that a predictive model is built from inconsistent data, conflicting outcomes will make the model difficult to achieve an accurate estimation.

We study 4 effort estimation data sets from the PROMISE software repository [30]. The result shows that *CIL* is able to evaluate the quality of a data set in terms of its consistency. Furthermore, our findings reveal an interesting application of the metric that is it can be used as a measurement to predict the possibility that an effort model built from a data set can produce high accuracy. *CIL* can be used in such a way

EASE '15 April 27 - 29, 2015, Nanjing, China

http://dx.doi.org/10.1145/2745802.2745820.

because that it is difficult to build an accurate estimation model from extremely inconsistent data samples.

The rest of the paper is organized as follows: Section 2 overviews the ESE data and the data inconsistency issues. Section 3 explains our proposed *CIL* metric. Section 4 and 5 explain the experimental setup and the results, respectively. Section 6 discusses the proposed *CIL* metric in detail including future opportunities. Finally, we conclude the paper in Section 7.

2. BACKGROUND

There are many existing definitions of data quality in software engineering research. In this study, we follow a widely accepted definition defined by Liebchen and Shepperd as "being fit for purpose" [9, 27]. Liebchen and Shepperd have pointed out that software engineering researchers often have a lacks of interest in addressing the issue of data quality. Even if a research conclusion is highly dependent on the quality of the data that were used in the study, Liebchen and Shepperd also have reviewed hundreds of software engineering research papers and found only 23 of which have discussed or addressed a problem regarding the data quality. To the best of our knowledge, there is no industrial standard practice to evaluate the data quality. The literatures have only emphasized guidelines for improving the selection of data sets during the data collection process, such as the pioneer study by Basili and Weiss in 1984 [4].

2.1 ESE Data Sets for Effort Estimation

According to the "being fit for purpose" definition [9, 27], the "purpose" being focused in this study is to build an accurate effort estimation model by focusing only on using software engineering data. This is unlike a qualitative approach that commonly uses expert opinions [16], the approach presented here provides a quantitative way to to tackle the problem. We assume that we only have data sets to be used as information sources.

Table 1 shows an example data set named Kemerer data set, it is a data set from the PROMISE repository [30] and commonly used for related studies. Data sets are recorded in many organizations for their software development process and are commonly used for the project planning [39]. Each row of Table 1 represents one single software project, and each column represents project features such as development size (e.g. KSLOC and AdjFP), and development man-hours (e.g. EffortMM). These features are also commonly seen in other PROMISE data sets [30].

The highlighted rows in Table 1 show an example of an inconsistent pair of project data. When we look closer at the project id 9 and id 11, the two projects seem to be very similar to each other in terms of size, but the development effort of the project id 11 is about two times greater than that of the project id 9. According to this conflict, it is difficult to build an accurate estimation model from this data set, even if a feature selection technique is utilized in prior to build the model. We are studying this behavior in this paper.

2.2 Inconsistent Data Issue in ESE Data Sets

Inconsistent data is the data that can not be easily explained [26], so that the interpretation of these data is problematic during a predicting model task. For example, in binary classification, it is not logical to train a classifier with the same training feature set with labels in different classes. Even if inconsistency is one of the most severe problems they exist in data sets commonly used [7, 8, 37, 41], there is no existing guidance nor a procedure for researchers or practitioners to properly handle this issue.

Table 1: An example of software engineering data sets (Kemerer data set from [27])

ID	Language	Hardware	Duration	KSLOC	AdjFP	EffortMM
1	1	1	17	253.6	1217.1	287
2	1	2	7	40.5	507.3	82.5
3	1	3	15	450	2306.8	1107.31
4	1	1	18	214.4	788.5	86.9
5	1	2	13	449.9	1337.6	336.3
6	1	4	5	50	421.3	84
7	2	4	5	42	99.9	23.2
8	1	2	11	200	993	130.3
9	1	1	14	289	1592.9	116.0
10	1	1	5	39	240	72
11	1	1	13	254.2	1611.0	258.7

The lack of a procedure to identify inconsistent data motivates us to develop a novel software metric for evaluating the inconsistency level of ESE data sets. In this work, we propose a software metric based on an analysis between all the possible paring of the project cases, and to quantify the level of inconsistency of software development project data.

According to our survey, we found no existing software metric that can directly quantify the level of consistency in ESE data sets. Researches in software engineering area that discussed consistencies was only focused on measuring consistency between studies [10, 15, 28]. For example, Mair and Shepperd [28] analyzed 67 journal papers and 104 conference papers in effort estimation and confirmed that the lack of consistency in those studies. The procedures used in [28] are to construct an analysis framework and manually derive the aggregated result to be the final result, thus there was no standard measure used in their study.

In systematic reviews, one of the standard measures commonly used to quantify consistency between studies is Cochrane's I^2 measure [14]. However, we found no study in our area using this measure. Kitchenham [20] discussed measures in the Cochrane family that they are suitable only in the situation where the amount of data to analyze is a sufficiently large, such as in medical. We investigated the mathematic theory behind I^2 and found that its statistic properties cause the metric inapplicable for testing small sample size data sets. Unfortunately, most of available ESE data sets used in predictive modelling are relatively small.

3. CONSISTENCY BETWEEN SOFTWARE PROJECT CASES

Inconsistent data is one of the primary reasons for predictive model failing to achieve an accurate estimate [38]. This is because it is difficult to build an effort model that can explain a data set containing conflicting project cases. In this study, we propose a software metric named Cases Inconsistency Level (*CIL*) as a software metric used for analyzing conflicts in the predictive modeling data set.

Since software engineering data collected for a different purpose have different types of conflict, in this study we use only software effort estimation data to present the notion of our proposed metric. Software effort estimation is a good example that can be applied for many other software engineering problems employing predictive modeling techniques. To explain in more detail, effort estimation models the software project feature (x_1, x_2, ..., x_n) of a data set to derive the target variable y, which is the software development effort.

3.1 The Proposed *CIL* Metric

We define a "consistent data set" as a data set comprising of no project case that conflicts to any other cases. In the case of effort estimation data set, used in this study, any of the two project cases will be considered inconsistent if:

- The two cases are very similar in project features, but their effort values are significantly different.
- The two cases are significantly different in project features, but their effort values remain very close to each other.

The principle idea of *CIL* is to analyze all the possible pairing of project cases in a data set and to count the number of conflicting pairs. The *CIL* metric is calculated as follows:

$$\begin{aligned} CIL &= \frac{N(pp_{consistent})}{N(pp)} \text{ where} \\ pp_{consistent} &= (p_i, p_j) \mid \\ &\quad (p_i, p_j) \in pp' \wedge EffortDiff(p_i, p_j) \geq 1.00 \\ &\quad \cup (p_i, p_j) \in pp'' \wedge EffortDiff(p_i, p_j) < 1.00 \\ pp' &= (p_i, p_j) | (p_i, p_j) \in pp \wedge Distance(p_x, p_y) < \alpha \\ pp'' &= (p_i, p_j) | (p_i, p_j) \in pp \wedge Distance(p_x, p_y) \geq \alpha \end{aligned} \quad (1)$$

where α is a threshold in the interval between 0 and 1, used for observing the extreme pairs of the project cases that are greatly different in project features.

Since consistency is observed in relativity, we measure the difference of the effort values in each project pair by calculating a relative difference between any of the two effort value pairs as follow:

$$EffortDiff(p_i, p_j) = \frac{Effort(p_i) - Effort(p_j)}{\frac{Effort(p_i) + Effort(p_j)}{2}} \quad (2)$$

where (p_i, p_j) is a pair of project cases, and $Effort(p_i)$ is an actual effort value used for developing p_i. We measure the difference between project features in each pair using a distance function called Interpolate Value Difference Metric *(IVDM)* [42]. Comparing to the conventional euclidian distance function, this *IVDM* function utilizes more information that is based on the correlation between a pair of any feature and the target variables. *IVDM* was also proven to be a better function to be used for calculating the similarity between possible pairing of project cases than the conventional euclidian distance function, because *IVDM* can capture more information from the target variables [3, 5, 34].

IVDM is based on the calculation of the difference in conditional probabilities between pair of project features, and is calculated as follow:

$$IVDM(X, Y) = \sum_{i=1}^{m} \sum_{j=1}^{n} (P(c_j|x_i)' - P(c_j|y_i)')^2 \quad (3)$$

where x_i and y_i are elements in vectors of project cases with value a, c is the class describes the target variable which is calculated by discretizing the effort variable, m is the number of project features, and n is the number of classes of the discretized effort variable. $P(c_j|x_i)'$ is a probability that a project case with an effort value in class c_j which has a project feature x with a value a. This probability is estimated by counting the number of (c_j, x_i) that have been appeared out of the total number of x_i.

For continuous variables, *IVDM* adjusts the probability estimation by using interpolation to retain the important information available in the continuous values. In the case of continuous variables, the probability $P(c_j|x_i)$ is adjusted as follows:

$$\begin{aligned} P(c_j|x_i)' &= P(c_j|x_i) + (P(c_j|x_{i+1}) \\ &\quad - P(c_j|x_i) \times \frac{x_i - mid(x_i)}{mid(x_{i+1}) - mid(x_i)}) \end{aligned} \quad (4)$$

where x_{i+1} is the next discretized class of x_i (i.e. the values in feature x that are discretized in to a class with a value greater than x_i by 1), $mid(x_i)$ and $mid(x_{i+1})$ are the medians of the discretized class x_i, and x_{i+1} respectively.

Figure 1 presents two scatter plots showing examples of a low quality and a high quality data set. The lower quality data set, as shown on the left side, has higher number of project case pairs falling in the threshold area of $\alpha = 0.3$. According to the *CIL* value calculated, we assume that the data set on the right has higher quality in terms of data set consistency. For example, when a pair of project cases has a very small *IVDM* distance, this pair will be a consistent pair of the two project cases if the relative difference of effort values is also very small.

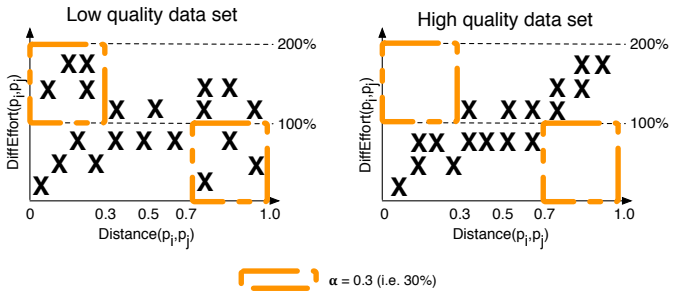

Figure 1: Examples of low quality and high quality data sets according to the definition used in this study

4. METHODOLOGY

Based on the basis that a more consistent data set is more likely to contribute to a more accurate prediction model, we evaluate the *CIL* metric by observing the correlation between (1) *CIL* value and (2) the estimation errors of multiple estimation models built from the experimental data sets. We

Table 2: Experimental data sets

Data sets	#Cases	#Features	Pre-processors	#Selected features	Selected features
Kemerer	15	7	NONE	5	Language Hardware KSLOC AdjFP RAWFP
			SFS	5	Language Hardware KSLOC AdjFP RAWFP
			SWR	1	AdjFP
			PCA	3	KSLOC AdjFP RAWFP
Miyazaki	48	9	NONE	7	KLOC SCRN FORM FILE ESCRN EFORM EFILE
			SFS	7	KLOC SCRN FORM FILE ESCRN EFORM EFILE
			SWR	1	EFILE
			PCA	5	FORM FILE ESCRN EFORM EFILE
China	499	12	NONE	10	AFP Input Output Enquiry File Interface Added Changed Deleted Resource
			SFS	9	AFP Input Output Enquiry File Interface Added Deleted Resource
			SWR	1	Added
			PCA	6	File Interface Added Changed Deleted Resource
Nasa93	93	24	NONE	21	projectname cat2 forg center mode rely data cplx time stor virt turn acap aexp pcap vexp lexp modp tool sced equivphyskloc
			SFS	21	projectname cat2 forg center mode rely data cplx time stor virt turn acap aexp pcap vexp lexp modp tool sced equivphyskloc
			SWR	2	forg equivphyskloc
			PCA	11	virt turn acap aexp pcap vexp lexp modp tool sced equivphyskloc

expect that the data sets with lower *CIL* values will be on average have a lower estimation error than that of the data sets with higher CIL values.

In the experiment, we observed the correlation between *CIL* and the estimation accuracy using public industrial data sets. Three different α values: $\alpha = 0.1$, $\alpha = 0.2$, and $\alpha = 0.3$ were given to be the thresholds to observe the extreme pairs of the project cases pairing. We observed multiple α, because we assumed that the different α values implies different level of inconsistency.

A data set with high $\alpha = 0.1$ is a data set consisting of more number of project cases that are extremely inconsistent. Therefore we assumed that CIL with $\alpha = 0.1$ can be used as an internal metric. *CIL* with $\alpha = 0.1$ can be a metric to indicate whether or not a data set is critically inconsistent. In the case that a data set consists of many project case pairs falling in $\alpha = 0.1$ threshold area, then an estimation model built from that data set is more likely to be an inaccurate model. On the other hand, we assumed that both $\alpha = 0.2$ and $\alpha = 0.3$ are better to be used as external metrics to compare a data set with other data sets.

4.1 Experimental Data Sets

We used four real-world data sets to evaluate our proposed *CIL* metric. All the four data sets are available in the PROMISE repository (accessible through [30]). The PROMISE repository is where the empirical software engineering datasets were collected and made available for software engineering researchers and practitioners to tackle challenges in software engineering problems.

It is typically that some of the feature variables are omitted from the estimation, because these variables can not contribute to the target variable. If these variables are included in the predictive model, the estimation accuracy will be reduced. To reduce the negative influence from the irreverent variables, feature selection is commonly applied in the preprocessing to remove these variables. Building an effort estimation model, *Do nothing (NONE)*, *Sequential forward selection (SFS)*, *Stepwise regression (SWR)*, and *Principle component analysis (PCA)* are the variable selection methods commonly applied in the studies [1, 17]. We use these methods in our experiments to observe numerous types of data sets.

We observed the data sets that were preprocessed with a different methods to be different instances of data set. Table 2 summarizes the properties of all 16 data set instances used in our study. Note that the *NONE* method also removed the dependent variables that are dependent on the effort. These variables could not contribute to the estimation of the effort value, because they are not available before the actual effort value is known. From Table 2, *SFS* selected as much features as they were selected by *NONE*, followed by *PCA*. Finally, *SWR* selected the least number of features.

4.2 Effort Estimators

To select only accurate effort estimation models for evaluating the *CIL* metric, we followed Keung et al. [17] who advocated a relative ranking of the 90 prediction models commonly used as software effort estimators. This ranking was constructed by performing over 10,000 comparisons of the 90 models. Therefore, the ranking is stable enough to be directly used as a source for selecting the accurate estimators for our study.

Table 3: The selected effort estimators

Rank	NONE	SFS	SWR	PCA
1	CART(yes)	CART(yes)	CART(yes)	PLSR
2	CART(no)	CART(no)	CART(no)	PCR
3	PLSR	ABE0-5NN	ABE0-1NN	CART(yes)
4	PCR	ABE0-1NN	ABE0-5NN	CART(no)
5	ABE0-1NN	LReg	PCR	SWR

According to [17], the ranking also differed by the different feature selection methods used in the preprocess. Thus, we used different estimators for the data sets with different processing techniques. The selected estimators were the top-5 of the ranking list, of which were separately selected for each different preprocessor. Table 3 summarizes the estimators that were selected for our evaluation.

The *CART* (classification and regression trees) technique is on average the best estimator of the 4 preprocessing methods. *CART(yes)* is a *CART* model which the generated tree is pruned into a tree with lowest error rate using cross-validation [24]. *CART(no)* is a *CART* model that uses the full tree. *PLSR* (partial least squares regression), *PCR* (principal components regression), *SWR* (stepwise regression), and *Leg* (linear regression) are a variety of regression models. *PLSR* and *PCR* are fit to the PCA processing technique, but

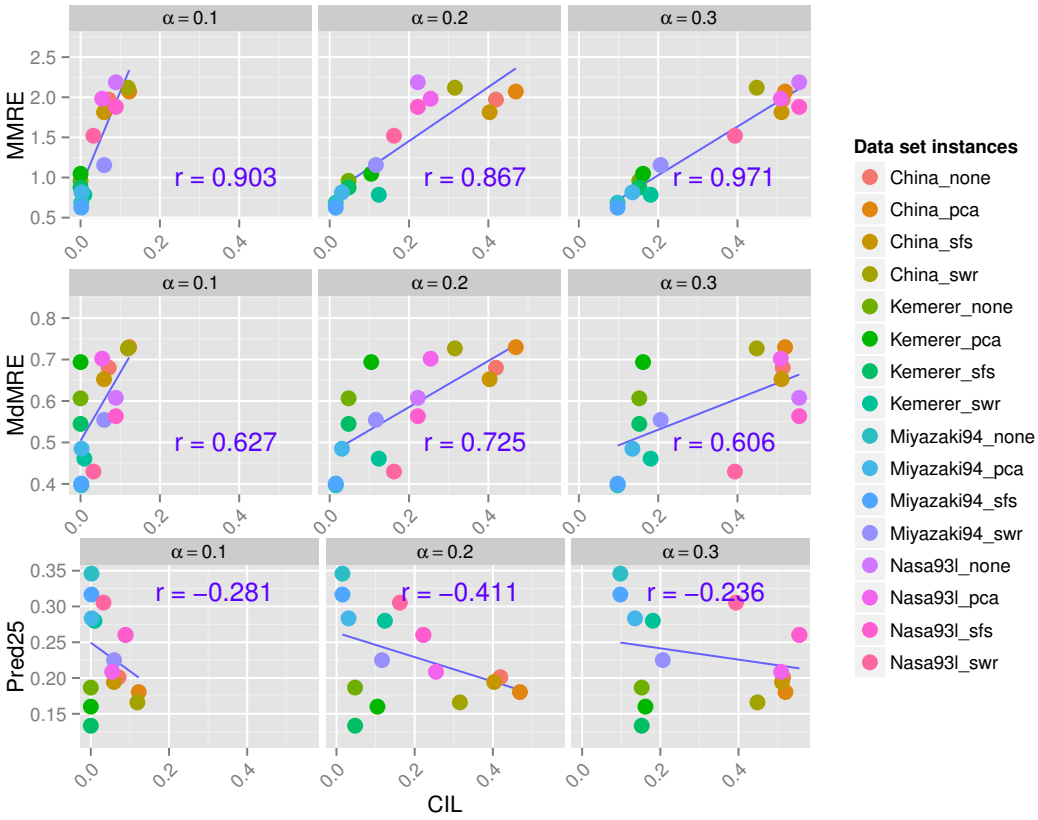

Figure 2: The correlation between *CIL* and the estimation errors

they seem to be not making the best use of the *SFS* and the *SWR* techniques. *ABE0-1NN* and *ABE0-5NN* are the standard setup of analogy-based estimation. *ABE0-1NN* uses only the closest analog to predict the effort value, whereas *ABE0-5NN* uses and adapts the 5 closest analogues.

$$MMRE = \sum_{i=1}^{n} MRE_i$$

$$MdMRE = median(MRE_{1->n})$$

$$PRED(25) = \frac{100}{N} \sum_{i=1}^{N} \begin{cases} 1 & if \quad MRE_i \leq 0.25 \\ 0 & otherwise \end{cases}$$

$$\text{where} \qquad MRE_i = \frac{|Actual_i - Predict_i|}{Actual_i} \tag{5}$$

To make the estimation results consistent with Keung et al. [17], we sampled the train and test sets from the data sets using leave-one-out method. The leave-one-out separates a project case to be used as a test case and trains the model with the remaining project cases. The estimation accuracy is achieved by averaging all the errors estimated from all tests. Error measures we used in our experiments are *MMRE* (The mean magnitude of relative error), *MdMRE* (The median magnitude of relative error) and *Pred(25)*. The three error measures are continually used as standard error measures in the literatures in area of software effort estimation. [2, 13, 17, 22]. Equation 5 depicts the functions of *MMRE*, *MdMRE* and *Pred(25)* respectively.

5. RESULTS

5.1 Evaluating the *CIL* Metric

Figure 2 presents the linear correlation (r) between *CIL* and the estimation accuracy. Three columns represent the three different α values. Three rows of Figure 2 represent the three error measures. In each single graph in Figure 2, the x-axis represents the *CIL* values, and the y-axis represents the estimation error. A lower $MMRE$ and $MdMRE$ value means a better accuracy, whereas a higher $Pred(25)$ means a better accuracy. Each single point is the mean error of the estimated effort tested by a data set instance. The errors were aggregated from the 5 most accurate estimators differing by different preprocessors.

By using Pearson correlation coefficient (r), the three columns of Figure 2 show that the correlation strength varied among different error measures being used. For all the three α values setup, the results show that *CIL* and $MMRE$ was greatly correlated. Also the correlation strength was considerably high when measuring with $MdMRE$. All the three α setups were achieve an r with values greater than 0.6. This means that an analysis of a data set using the *CIL* metric is definitely applicable to imply the estimation accuracy when measuring with either $MMRE$ or $MdMRE$.

It was slightly different for $Pred(25)$. Only $\alpha = 0.2$ showed a correlated result. We believed that the difference was caused by the mechanism of the $Pred(25)$ measure itself, that is the $Pred(25)$ does not measure the magnitude of the total errors, but it only measures a number of project cases that are relatively more accurate than the other cases in percentage. Therefore, this caused a weaker correlation strength when measuring with $Pred(25)$ than that of when

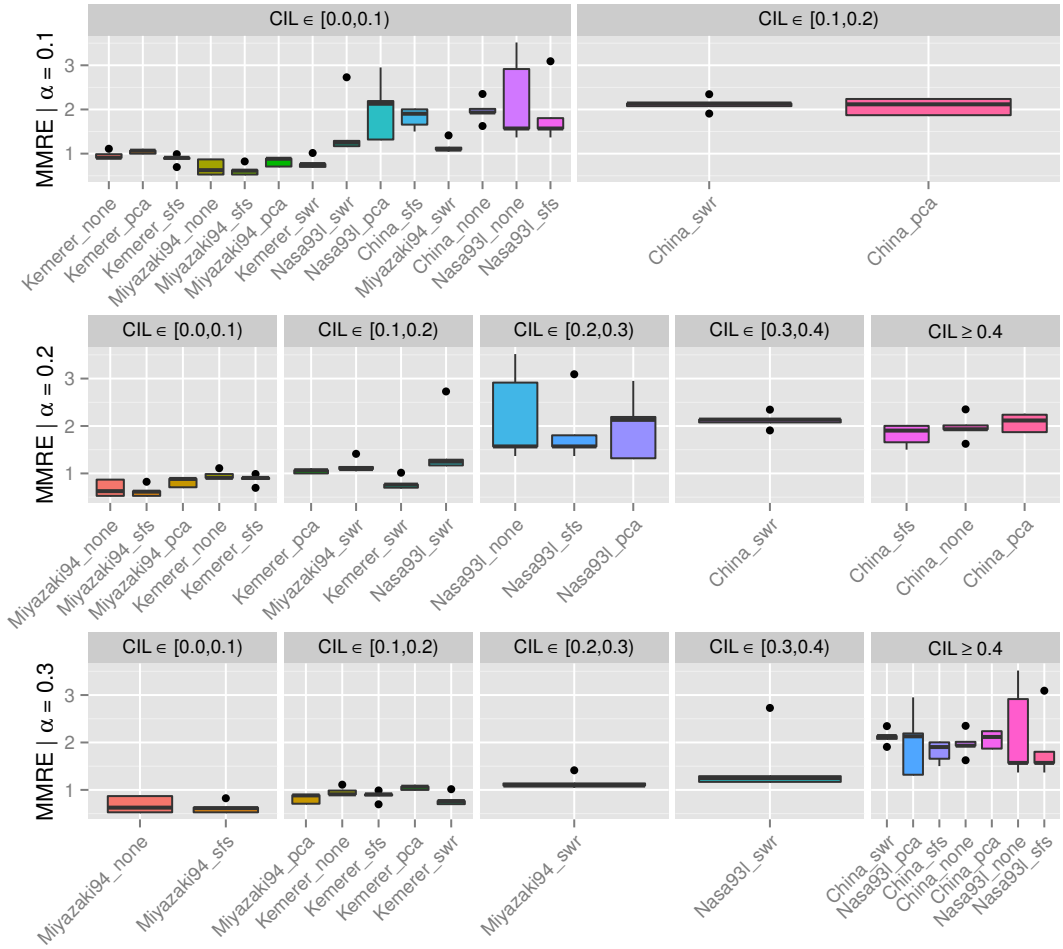

Figure 3: Analyzing experimental data sets by observing *CIL* in multiple ranges

measuring with $MMRE$ and $MdMRE$. Nevertheless, the r value of 0.4 was still high enough to indicate that CIL with $\alpha = 0.2$ and $Pred(25)$ was correlated.

5.2 Analyzing Experimental Data Sets using the *CIL* Metric

Figure 3 shows box plots of the $MMRE$ errors observed with the three different α values. Each column of Figure 3 divides the data sets by the CIL values. We binned the CIL values into five ranges: $[0.0, 0.1)$, $[0.1, 0.2)$, $[0.2, 0.3)$, $[0.3, 0.4)$, and $[0.4, \infty)$. These five bins were used to discuss the consistency level of the 16 data set instances.

Let $CIL@\alpha = 0.1$ defines the CIL value of a data set with the α threshold set to 0.1. Focusing on $CIL@\alpha = 0.1$ in Figure 3, the China_pca and China_swr instances were the only instances with CIL value higher than 0.1. As presented in the box plot, the best accuracy that the two instances could achieve (i.e. the lower bound of the box plots) were less accurate than that of all the other data sets. This result strongly supported our assumption that $CIL@\alpha = 0.1$ can be a metric to identify an extremely inconsistent data set.

The results based on $CIL@\alpha = 0.2$ added China_none and China_sfs to be the inconsistent data set instances. The best estimation accuracy that these two instances could achieved were just slightly better than that of the China_pca and China_swr data set instances. The result empirically showed that a data set with $CIL@\alpha = 0.2$ value higher than 0.2 had a worse estimation accuracy than that of the data set instances with the value less than 0.2.

For $CIL@\alpha = 0.3$, the result showed that data sets with $CIL@\alpha = 0.3$ value higher than 0.4 had a worse estimation accuracy than that of the other data sets. If we focus only on $CIL@\alpha = 0.3$, Miyazaki_none could be another candidate for being an inconsistent data set instance. However, this data set instance had a relatively low $CIL@\alpha = 0.2$, where the result showed that this data set instance can achieve an accurate estimate. As it was assumed in the previous section, this case thus can be an example to demonstrate that $CIL@\alpha = 0.2$ and $CIL@\alpha = 0.3$ should be used together when determining a consistency level of a data set. The value of $CIL@\alpha = 0.3$ is additionally useful in identifying a consistent data set. As shown in Figure 3, Miyazaki94_none and Miyazaki_sfs are the data set instances with $CIL@\alpha = 0.3$ less than 0.1. Comparing the the other data set instances, The Miyazaki94_none and Miyazaki_sfs instances are the only two instances that achieved the minimum $MMRE$ lower than 0.5, whereas their worst $MMRE$ achieved by the 5 estimators were less than 1.0.

The results from these first two experiments show that less accurate prediction is obtained by a data set with high CIL

value: $CIL@\alpha = 0.1$ value greater than 0.1, $CIL@\alpha = 0.2$ value greater than 0.2, and $CIL@\alpha = 0.3$ value greater than 0.4. Therefore, the results enable us to say that CIL is applicable to predict whether or not an effort model built from a data set can achieve high accuracy.

5.3 On a Possibility of Aggregating *CIL* with Multiple α to be a Combined Metric

According to our results, our proposed *CIL* with multiple α setups can be a useful quantifier for the consistency level of a data set. In future applications of the metric, such as an ensemble procedure [22] to identify the overall quality of a data set, it may require the only single *CIL* value that can describe the overall consistency level. Different from the *CIL* with multiple α setups, we are aware that none of a single selected α value could be a representative of the *CIL* metric. For example, using only $CIL@\alpha = 0.1$, the metric can reasonably indicate only a few extremely inconsistent data sets. On the other hand, using only $CIL@\alpha = 0.3$ without considering it with $CIL@\alpha = 0.2$, the result based on the Miyazaki_none data set instance shows that the metric was able to indicate a consistent data set, but it might fail to distinguish between severely inconsistent data sets and normal data sets.

And since there is no single best α parameter value, we assumed that there may be best combination of the parameter values. We explored a possibility to aggregate the three α values used in our experiments to be a single-value numeric *aggregated CIL* metric. We examined two simple procedures that are commonly used when aggregating the effort values produced by multiple estimators [22,29]. The two procedures are to calculate the (1) unweighted arithmetic mean and (2) inverse-ranked weighted mean (i.e. $IRWM$ [29]) of the *CIL* with the three α parameters.

Note that our aim of this discussion is not to investigate complex combination procedures, but only to explore the possibility of utilizing *aggregated CIL* metric. We are aware that computing the simple arithmetic mean of individual CIL with multiple α setups might be unreliable [32]. We thus include $IRWM$ in this experiment. According to [32], $IRWM$ could also introduce spurious problems when the applied weight is not clear. However, following a study of ensemble method in [22], where the applied weight is reasonable and $IRWM$ was proved successful, we believe that the weight to be applied here is reasonable enough to represent an *aggregated CIL* metric. Here, the $IRWM$ of the *CIL* with three different α values is $(\sum_{i=1}^{3} i \times \alpha_i)/(1+2+3)$. More generally, $IRWM$ applies a ranking of a single value to imply the importance of this value. And since we already discussed that the $CIL@\alpha = 0.1$ is the most important, we used $IRWM$ to assign more weight to $CIL@\alpha = 0.1$.

Figure 4 shows that the two simple combination procedures can be used for summarizing the *CIL* with multiple α values. The correlation strength between *CIL* and the estimation accuracy in each graph is sufficient to infer causation. Comparing the two combination procedures, $IRWM$ was more robust than the unweighted mean. Even though $IRWM$ gave a slightly worse correlation strength between CIL and $MMRE$, it performs better for $MdMRE$ and $Pred(25)$. Note that no single α parameter value could describe both $MdMRE$ and $Pred(25)$ at the same time. Therefore, this result supports our assumption that (1) CIL with $\alpha = 0.1$ is the most important setup, and (2) combining of the parameter values will result in better models.

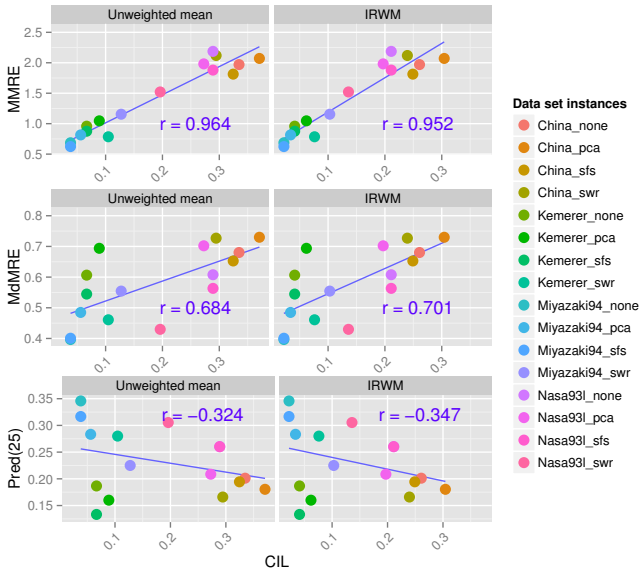

Figure 4: The correlation between the aggregated *CIL* and the estimation errors

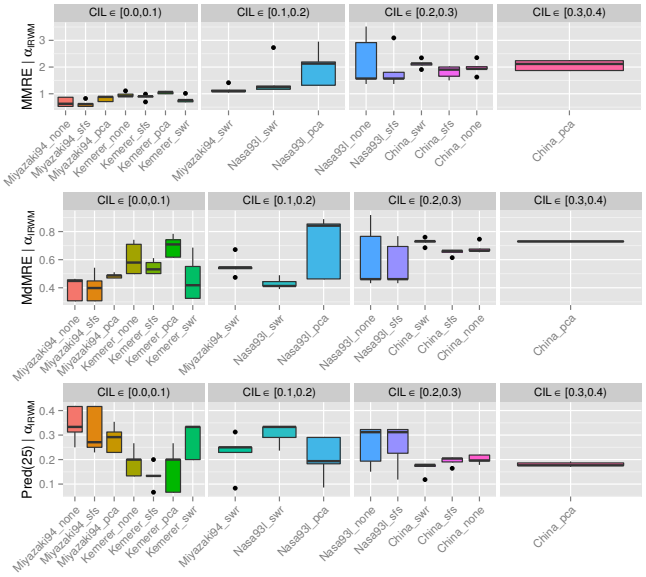

Figure 5: Discussing the aggregated *CIL* in multiple range

Figure 5 shows box plots of the three error measures observing the *aggregated CIL* metric using $IRWM$. The *aggregated CIL* metric was clearly applicable to predict the estimation accuracy in terms of $MMRE$. The estimation over the data set instances with CIL less than 0.1 had a minimum $MMRE$ less than 1, and the instances with CIL higher than 0.3 were relatively less accurate then the other data set instances. The results were also clear for the $MdMRE$ and $Pred(25)$ that the instances with CIL higher than 0.3 were relatively less accurate than the others.

6. DISCUSSION AND FUTURE WORK

This study introduces a software metric that can empirically quantify the level of inconsistency of a data set. Comparing with the other conventional metrics that are commonly used for identifying the data set quality, we believe that the proposed *CIL* metric is (1) more robust and (2) is a suitable metric to predict if an accurate estimation model can be built from the selected data set.

The robustness of *CIL* is obtained primarily by the use of *IVDM* distance function. For example, $Project_A$ with (large size, high-level language) selected as features and $Project_B$ with (medium size, low-level language) seem to require similar effort values. In this case, the conventional euclidean distance is unable to capture the similarity between these two projects, because the function will assign a maximum distance to *Language*, and thus considers the two project cases as completely dissimilar. Unlike the euclidean distance, information provided by the effort value enables *IVDM* to capture the similarity between $Project_A$ and $Project_B$. Therefore, using *IVDM* as a core function of *CIL* provides a better handling to the coincidental issue.

We evaluated the *CIL* metric over multiple instances of data sets. A test in this way allows us to further discuss the effect of feature selection on eliminating inconsistency. Table 3 shows that data set instances preprocessed by SFS and SWR were tested by mostly the same set of estimators. However, the results in Figure 3 shows that the instances from the same data set that were differently preprocessed by the two methods (e.g. Miyazaki94_sfs vs Miyazaki94_swr in the $MMRE|\alpha = 0.3$ plot) did not result in the same $MMRE$ range. This evidence shows that feature selection does affect the level of inconsistency of a data set. However, to the best of our knowledge, there is no existing objective function that selects a feature subset with an optimized level of consistency. Since our results show that a set of features with the lowest level of inconsistency will offer the best prediction accuracy, we suggest an interesting future work to explore the CIL metric in the feature selection process.

From the experiment that evaluated the *aggregated CIL* metric, we recognized that the metric has a long way forward to finally optimize as an ensemble metric. From Figure 5, using $IRWM$ was considered inapplicable to predict the estimation accuracy in terms of $MdMRE$ and $Pred(25)$ for a data set with low CIL values. Therefore, future work is required to further investigate more complex combination schemes to better combine multiple α values.

6.1 The Applications of *CIL* in Combination with Other Data Cleaning Techniques

Excessive missing data [31] can produce a negative effect on the performance of a predictive model. If we omit numerous records with missing values, a predictive model may be built with a lack of data samples. This can lead to an undertrained model whereby the model will also lack the general predictive power [6]. Multiple imputation techniques such as *Bayesian multiple imputation* and *regression* [35] have been commonly used in order to handle missing values [40]. Imputation is a technique to guess and fill in the missing values with one or more plausible values [18].

The *CIL* metric could improve the imputation process. The candidates of the imputed values suggested by multiple imputation techniques can be tested by *CIL* without a need of executing the entire prediction model. Thus, *CIL* can contribute to shorten the process in practice, the use of *CIL* in this way can reduce the total processing time spent on imputing the missing values.

Noisy data is the data samples that cannot be interpreted, due to a large discrepancy. Compared to missing values, noise is more difficult to handle. A noise detection process is necessarily employed before noisy data can be corrected [25]. According to [6], noise detection and correction are still a long way for reaching a successful level. This is because there are many variations in noise characteristics, and it is impossible to identify a single technique that can fit for all types of noise.

Some imputation techniques that are effective for handling missing values were also proved to be very useful for noise detection and correction [6, 35]. Therefore, the *CIL* metric can also contribute to this problem.

Outlier value is a data quality issue that is commonly discussed as a first step to analyze the quality of data. Outliers can cause problems of the skewed distributions. This problem often produces spurious effect to statistical analysis, such as in correlation test. Similar to measuring how complete a data set is, a data set containing no outlier value also does not imply that it can achieve high accuracy. An integration between the existing outlier removal techniques and the use of *CIL* metric is an interesting future work for us. For example a software project that was developed in an extremely different scale can present extreme values in either software size or development effort. The use of *CIL* metric has the potential to simply overcome this issue commonly seen in the outlier removal process.

Consistency and diversity are issues that are somewhat incompatible with to each other. Removing all the inconsistent project data may lead to a lack of diversity, and the resultant data set may insufficiently reflect the real world. Therefore, we may require further consideration in order to cope with the two issues. Nagappan et al. [33] proposed a data sample coverage metric, defined as the percentage of projects in a population that are similar with respect to a given sample to determine coverage and diversity. The sample coverage metric enables the assessment of the quality of the given sample, on the hypothesis that the higher the coverage the better the prediction outcome in terms of its dispersed diversity. The trade-off between the diversity and the dataset consistency must be handled with extra care, if an increased diversity is derived from data points from largely dispersed samples, then the prediction performance will be influenced. As a future work, we are planning to incorporate metrics such as sample coverage [33] in our ongoing study to develop a more comprehensive approach to balance the needs of both diversity and consistency of the given data samples.

6.2 Towards Quality Improvement of ESE Data Sets

There are many reasons of data inconsistency in ESE data sets. In our view of quantitative software engineering, we strongly believe that the quality of ESE data set can be greatly improved by a better data collection procedure. We found a lack of meta-information that can adequately describe the data sets used in this study. Without such meta-information, it seems to be impossible to know the actual level of quality of a data set. For instance, researchers considered the China data set is of very-low quality, because

the dispersed data points cannot be interpreted. Most of the data points are possibly correct, but they were an aggregation of numerous types of project subsets from different organization. Therefore, attempting in terms of improving data set size may be inappropriate with respect to modelling. If a company aims at building a single large data set, we suggest that the data set should be well described, at least, it should provide information showing which sub project the single project data came from.

We expect a better data collection procedure that enables us to build a well-defined structure for ESE data sets. The structured data such as protein sequence offers efficient and effective model for protein function prediction [23], though; the estimation technique commonly used is seen to be very similar to effort estimation (i.e. ABE0). Furthermore, data sets commonly used in ESE research do not provide information to understand the numerous changes that occur over time (e.g. technologies and people). Since prediction models rely heavily on historical records, a model will offer an accurate result as long as it is relevant and representative (e.g. future projects are similar to those undertaken in the past). This is nonetheless contributing to the industries where the data collection seems to be more organized and structure.

6.3 Threats to Validity

Although, a recent study has reported that ensemble of estimators will provide better accuracy than that of solo methods [22], we decide to demonstrate our test framework by using the consistently best solo methods in the pursuit of **external validity**. We recognized that the results are more generalized in such a way that we tested the *CIL* metric by using multiple solo estimators over multiple data set instances and preprocessed by multiple feature selection techniques.

Concerning **construct validity**, three error measures we used in our experiments were criticized as deprecated error measures [11, 21]. We recognized the deficiency of *MMRE*, but for easy interpretation and comparison, we used *MMRE* to demonstrate the results in the study, together with supplementary measures such as *MdMRE* and *Pred(25)*. Furthermore, the three error measures have been continually used as standard error measure to calibrate and validate effort estimators in the research area of effort estimation [2, 13, 17, 22].

7. CONCLUSION

Data set inconsistency is one of the most challenging problems that exist in any empirical software engineering (ESE) data set, because it is difficult to build an accurate estimation model from an inconsistent data set. This study proposes Cases Inconsistency Level (*CIL*), a software metric used for quantifying the level of inconsistency of ESE data sets. *CIL* analyzes all the possible paring of project cases in a data set and counts the number of conflicting pairs between the descriptive variables and the target variable. We evaluated multiple threshold values to observe the interplay between conflicting pairs and the quality of a data set. With a small threshold value, *CIL* can accurately identify extremely inconsistent data sets. Even if we used five most accurate estimation models selected separately regarding the different properties of the experimental data set instances, the instances with high *CIL* (i.e. low quality) achieved much worse accuracy than that were achieved by the other instances with lower *CIL* values. At higher threshold value, *CIL* metric performs better in identifying an consistent data set, where its ability to identify an inconsistent data set is reduced. Our result shows that the metric can feasibly identify either consistent data sets or inconsistent data sets. In additional to the ability to quantify the consistency level of a data set, *CIL* can predict if an effort model built from a data set can achieve higher estimation accuracy. Therefore, the *CIL* metric is sound and an important tool to evaluate and preprocess data sets for ESE research.

8. ACKNOWLEDGEMENTS

This study has been supported by JSPS KAKENHI Grant Number 26330086, and has been conducted as a part of the City University of Hong Kong research fund (Project No. 7200354, 7004222), Program for Advancing Strategic International Networks to Accelerate the Circulation of Talented Researchers, and the Global Initiatives Program for Promoting Overseas Collaborative Research Toward Graduate Education in Biological Sciences, Nano Science, and Information Technology, Nara Institute of Science and Technology.

9. REFERENCES

[1] M. Azzeh. A replicated assessment and comparison of adaptation techniques for analogy-based effort estimation. *Empirical Software Engineering*, 17(1-2):90–127, 2012.

[2] M. Azzeh and Y. Elsheikg. Learning best k analogies from data distribution for case-based software effort estimation. In *Proceedings of the 7th International Conference on Software Engineering Advances (ICSEA)*, pages 341–347, 2012.

[3] Y. Bao, N. Ishii, and X. Du. Combining multiple k-nearest neighbor classifiers using different distance functions. In *Intelligent Data Engineering and Automated Learning–IDEAL 2004*, pages 634–641. Springer, 2004.

[4] V. R. Basili and D. M. Weiss. A methodology for collecting valid software engineering data. *Software Engineering, IEEE Transactions on*, 10(6):728–738, 1984.

[5] E. Blanzieri and F. Ricci. Probability based metrics for nearest neighbor classification and case-based reasoning. In *Case-Based Reasoning Research and Development*, pages 14–28. Springer, 1999.

[6] M. F. Bosu and S. G. MacDonell. A taxonomy of data quality challenges in empirical software engineering. In *Proceedings of the 22nd Australian Conference on Software Engineering (ASWEC)*, pages 97–106, 2013.

[7] M. L. Brown and J. F. Kros. Data mining and the impact of missing data. *Industrial Management & Data Systems*, 103(8), 2003.

[8] J.-S. Chen and C.-H. Cheng. Software diagnosis using fuzzified attribute base on modified mepa. In *Proceedings of the 19th International Conference on Advances in Applied Artificial Intelligence: Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE)*, pages 1270–1279, 2006.

[9] P. B. Crosby. *Quality is free: The art of making quality certain*, volume 94. McGraw-Hill New York, 1979.

[10] T. Dyba and T. Dingsoyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9–10):833–859, 2008.

[11] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit. A simulation study of the model evaluation criterion mmre. *IEEE Trans. Softw. Eng.*, 29(11):985–995, 2003.

[12] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson. The misuse of the nasa metrics data program data sets for automated software defect prediction. In *Proceedings of the 15th Annual Conference on Evaluation and Assessment in Software Engineering (EASE)*, pages 96–103, 2011.

[13] S. Gupta, G. Sikka, and H. Verma. Recent methods for software effort estimation by analogy. *ACM SIGSOFT Software Engineering Notes*, 36(4):1–5, 2011.

[14] J. P. Higgins, S. G. Thompson, J. J. Deeks, and D. G. Altman. Measuring inconsistency in meta-analyses. *BMJ: British Medical Journal*, 327(7414):557, 2003.

[15] M. Jørgensen. A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, 70(1):37–60, 2004.

[16] M. Jørgensen and D. Sjøberg. *Expert Estimation of Software Development Work*. Wiley, 2006.

[17] J. Keung, E. Kocaguneli, and T. Menzies. Finding conclusion stability for selecting the best effort predictor in software effort estimation. *Automated Software Engineering*, 20(4):543–567, 2013.

[18] T. M. Khoshgoftaar and J. Van Hulse. Imputation techniques for multivariate missingness in software measurement data. *Software Quality Journal*, 16(4):563–600, 2008.

[19] S. Kim, H. Zhang, R. Wu, and L. Gong. Dealing with noise in defect prediction. In *Proceedings of the 33rd International Conference on Software Engineering (ICSE)*, pages 481–490, 2011.

[20] B. Kitchenham. Procedures for performing systematic reviews. *Keele University Technical Report TR/SE-0401*, 33(2004):1–26, 2004.

[21] B. Kitchenham, L. Pickard, S. MacDonell, and M. Shepperd. What accuracy statistics really measure. *Software, IEE Proceedings*, 148(3):81–85, 2001.

[22] E. Kocaguneli, T. Menzies, and J. W. Keung. On the value of ensemble effort estimation. *Software Engineering, IEEE Transactions on*, 38(6):1403–1416, 2012.

[23] L. Lan, N. Djuric, Y. Guo, and S. Vucetic. Ms-knn: protein function prediction by integrating multiple data sources. *BMC bioinformatics*, 14(Suppl 3):S8, 2013.

[24] R. J. Lewis. An introduction to classification and regression tree (cart) analysis. In *Annual Meeting of the Society for Academic Emergency Medicine in San Francisco, California*, pages 1–14, 2000.

[25] G. Liebchen, B. Twala, M. Shepperd, and M. Cartwright. Assessing the quality and cleaning of a software project dataset: An experience report. In *Proceedings of the 10th International Conference on Evaluation and Assessment in Software Engineering (EASE)*, pages 122–128, 2006.

[26] G. A. Liebchen and M. Shepperd. Software productivity analysis of a large data set and issues of confidentiality and data quality. In *Proceedings of the 11th IEEE International Software Metrics Symposium (METRICS)*, pages 46–48, 2005.

[27] G. A. Liebchen and M. Shepperd. Data sets and data quality in software engineering. In *Proceedings of the 4th International Workshop on Predictor Models in Software Engineering (PROMISE)*, pages 39–44, 2008.

[28] C. Mair and M. Shepperd. The consistency of empirical comparisons of regression and analogy-based software project cost prediction. In *Proceedings of 2005 International Symposium on Empirical Software Engineering (ISESE)*, page 10, 2005.

[29] E. Mendes, I. Watson, C. Triggs, N. Mosley, and S. Counsell. A comparative study of cost estimation models for web hypermedia applications. *Empirical Software Engineering*, 8(2):163–196, 2003.

[30] T. Menzies, B. Caglayan, Z. He, E. Kocaguneli, J. Krall, F. Peters, and B. Turhan. The promise repository of empirical software engineering data, June 2012.

[31] A. Monden. Some open problems in software project data analysis. In *Proceedings of the workshop on accountability and traceability in global software engineering (AIGSE)*, pages 11–12, 2007.

[32] K. Mordal, N. Anquetil, J. Laval, A. Serebrenik, B. Vasilescu, and S. Ducasse. Software quality metrics aggregation in industry. *Journal of Software: Evolution and Process*, 25(10):1117–1135, 2013.

[33] M. Nagappan, T. Zimmermann, and C. Bird. Diversity in software engineering research. In *Proceedings of the 9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*, pages 466–476. ACM, 2013.

[34] P. Phannachitta, J. Keung, A. Monden, and K. Matsumoto. Improving analogy-based software cost estimation through probabilistic-based similarity measures. In *Proceedings of the 20th Asia-Pacific Software Engineering Conference (APSEC)*, pages 541–546, 2013.

[35] N. Seliya and T. M. Khoshgoftaar. Software quality estimation with limited fault data: A semi-supervised learning perspective. *Software Quality Control*, 15(3):327–344, 2007.

[36] K. Strike, K. El Emam, and N. Madhavji. Software cost estimation with incomplete data. *Software Engineering, IEEE Transactions on*, 27(10):890–908, 2001.

[37] T. Tan, Q. Li, B. Boehm, Y. Yang, M. He, and R. Moazeni. Productivity trends in incremental and iterative software development. In *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement*, pages 1–10, 2009.

[38] M. S. Ted Friedman. Measuring the business value of data quality, 2011.

[39] M. Tsunoda, A. Monden, H. Yadohisa, N. Kikuchi, and K. Matsumoto. Software development productivity of japanese enterprise applications. *Information Technology and Management*, 10(4):193–205, 2009.

[40] J. Van Hulse and T. M. Khoshgoftaar. A comprehensive empirical evaluation of missing value imputation in noisy software measurement data. *Journal of Systems and Software*, 81(5):691–708, 2008.

[41] J. Van Hulse, T. M. Khoshgoftaar, C. Seiffert, and L. Zhao. Noise correction using bayesian multiple imputation. In *Proceedings of the IEEE International Conference on Information Reuse and Integration (IRI)*, pages 478–483, 2006.

[42] D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6(1):1–34, 1997.