Doctoral Thesis

Data Generation and Evaluation Methods for Mining Software Engineering Data Sets

GAN Maohua

Division of Industrial Innovation Sciences

Graduate School of Natural Science and Technology

Okayama University

March 2023

Abstract

Predictive data mining consists of five steps: setting objectives, data gathering, data preparation, applying data mining algorithms, and evaluating results. This thesis proposes a data generation method and two data evaluation methods for mining software engineering data sets that support data gathering, preparation and evaluation.

The first proposal is a method for artificially generating a “mimic” software project data set, whose characteristics are very similar to a given confidential data set. Instead of using the original (confidential) data set, researchers are expected to use the mimic data set to produce similar results as the original data set. To evaluate the efficacy of the proposed method, software development effort estimation is considered as potential application domain for employing mimic data. Estimation models are built from 8 reference data sets and their concerning mimic data. Our experiments confirmed that models built from mimic data sets show similar effort estimation performance as the models built from original data sets, which indicate the capability of the proposed method in generating representative samples.

The second proposal is a data quality metric called Similar Case Inconsistency Level (SCIL). Using SCIL, researchers can assess the quality of input data (training data) before conducting any data mining techniques. An empirical evaluation with 54 data samples derived from six large project data sets showed that SCIL can distinguish between consistent and inconsistent data sets, and that prediction models for software development effort and productivity built from consistent data sets can achieve relatively high accuracy.

The third proposal is a set of evaluation metrics called neg/pos-normalized accuracy measures to address the class imbalance issue in assessing performance of defect prediction models. The proposed measures enable researchers to compare defect prediction results across different data sets with different neg/pos ratios. A case study of defect prediction based on 19 defect data sets shows that the proposed measures enable us to provide a ranking of predictions across different data sets, which can distinguish between successful predictions and unsuccessful predictions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data mining, also known as Knowledge Discovery in Databases (KDD), is the field of discovering potentially useful information from large amounts of data [18]. To date, data mining has been applied in many fields such as retail, education, manufacturing as well as software engineering [69][94].

The two main objectives of data mining are prediction and description. Prediction involves using some variables or fields in the data set to predict unknown or future values of other variables of interest. Description focuses on finding patterns describing the data that can be interpreted by humans. Therefore, there are two types of data mining: predictive mining and descriptive mining with different functions and technologies [89, 41]. In this paper, we focus on the predictive mining of software engineering data such as software effort data sets and defect data sets.

As shown in Figure 1.1, data mining usually consists of five main steps: setting objectives, data gathering, data preparation, applying data mining algorithms, and evaluating results [41]. The first step, setting objectives, clearly stating the problem and setting clear, unambiguous objectives. At this stage, several hypotheses may be formulated, and a set of variables is specified. The second step, data gathering, concerns the collection and generation of data for use in data mining. The importance of data gathering can not be overstated, and data have been seen as the limiting factor for algorithmic development and scientific progress [35, 86]. The third step, data preparation, includes data preprocessing such as outlier detection, scaling, encoding, and feature selection. The fourth step, applying data mining algorithms, uses data mining techniques to build a data mining model, which is then applied

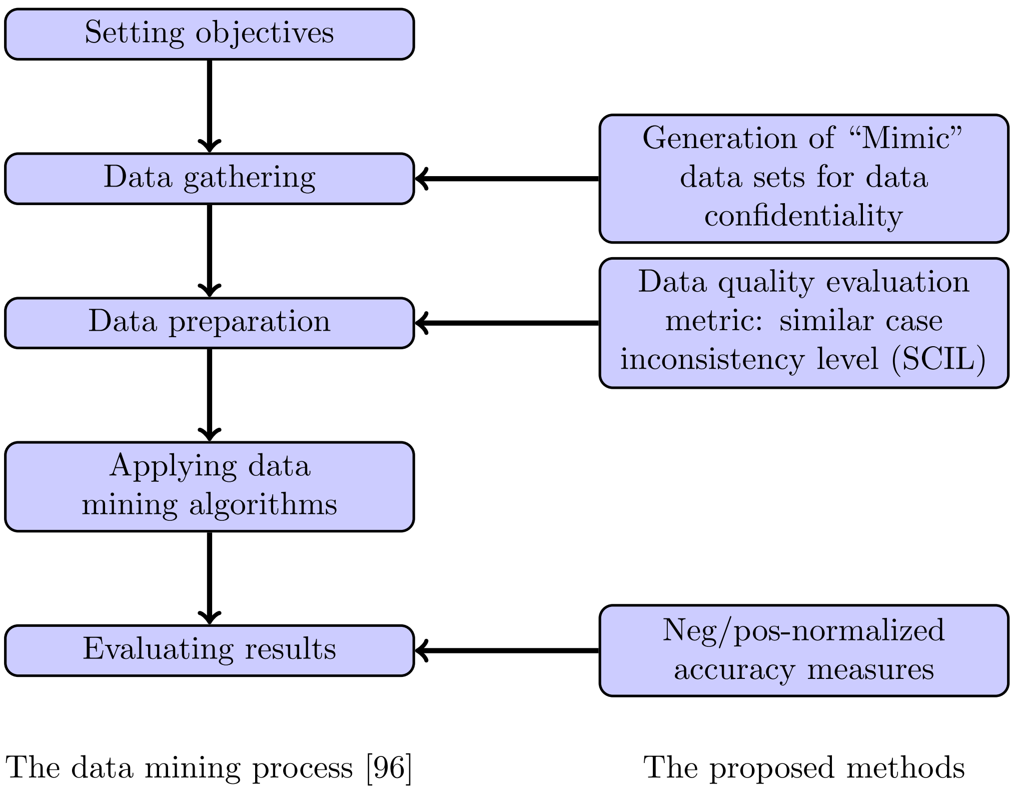

Setting objectives
Data gathering
Data preparation
Applying data mining algorithms
Evaluating results
Generation of "Mimic" data sets for data confidentiality
Data quality evaluation metric: similar case inconsistency level (SCIL)
Neg/pos-normalized accuracy measures
The data mining process [96]
The proposed methods

Figure 1.1 The data mining process and proposed methods.

to the prediction. The final step: evaluating results, gives evaluation and interpretation to the mining results. They should be valid, novel, useful, and understandable [41].

In Chapter 2 of this thesis, we focus on the step of data gathering in software engineering domain. To conduct empirical research on industry software development, it is necessary to obtain data of real software projects from industry. However, only few such industry data sets are publicly available; and unfortunately, most of them are very old [70, 20, 40, 3, 6, 77, 2, 49]. In addition, most of today's software companies cannot make their data open, because software development involves many stakeholders, and thus, its data confidentiality must be strongly preserved. To that end, Chapter 2 proposes a method for artificially generating a "mimic" software project data set, whose characteristics (such as average, standard deviation and correlation coefficients) are very similar to a given confidential data set. Instead of using the original (confidential) data set, researchers are expected to use the mimic data set to produce similar results as the original data set. The proposed method uses the Box-Muller transform for generating normally distributed random numbers; and exponential transformation and number reordering for data mimicry. To evaluate the efficacy of the proposed method, effort estimation is considered as potential application domain for employing mimic data. Estimation models are built from 8 reference data sets and their concerning mimic data.

Chapter 3 focuses on the step of data preparation. Software data sets derived from actual software products and their development processes are widely used for project planning, management, quality assurance and process improvement, etc. Although it is demonstrated that certain data sets are not fit for these purposes, the data quality of data sets is often not assessed before using them [56]. The principal reason for this is that there are not many metrics quantifying the fitness of software development data [58]. In that respect, this chapter makes an effort to fill in the void in literature by devising a new and efficient assessment method of data quality. To that end, we start as a reference from Case Inconsistency Level (CIL)[78], which counts the number of inconsistent project pairs in a data set to evaluate its consistency. Based on a follow-up evaluation with a large sample set, we depict that CIL is not effective in evaluating the quality of certain data sets. By studying the problems associated with CIL and eliminating them, we propose an improved metric called Similar Case Inconsistency Level (SCIL).

Chapter 4 focuses on the step of evaluating results. In evaluating the performance of software defect prediction models, accuracy measures such as precision and recall are commonly used. However, most of these measures are affected by neg/pos ratio of the data set being predicted, where neg is the number of negative cases (defect-free modules) and pos is the number of positive cases (defective modules). Thus, it is not fair to compare such values across different data sets with different neg/pos ratios and it may even lead to misleading or contradicting conclusions [65]. The objective of this chapter is to address the class imbalance issue in assessing performance of defect prediction models. The proposed method relies on computation of expected values of accuracy measures based solely on the value of the neg and pos values of the data set. Based on the expected values, we derive the neg/pos-normalized accuracy measures, which are defined as their divergence from the expected value divided by the standard deviation of all possible prediction outcomes. The proposed measures enable us to provide a ranking of predictions across different data sets, which can distinguish between successful predictions and unsuccessful predictions.

Finally, Chapter 5 provides a summary and discussion of this thesis as a whole.

Chapter 2

Generation and Evaluation of Mimic Software Project Data Sets

2.1 Introduction and motivation

Empirical software engineering relies to a great extent on real software development data. Namely, it is highly desirable to use data collected from industry software development projects. However, there exist only very few industry data sets, which are publicly available [70]. In addition, these data sets are quite old and have a small sample size, which pose a great problem in ensuring the validity and reliability of the research [20, 40, 3, 6].

As a matter of fact, most companies measure and accumulate data relating their own (recent) software development projects on an independent basis. However, companies cannot release any part of this (real) data due to two principal reasons. Namely, they need to comply to various data protection/privacy laws and standards. In addition, due to the large number of stakeholders involved, they are required to strictly preserve data confidentiality.

In this respect, this chapter proposes a method for artificially creating a data set with similar characteristics to a given industry data set. Namely, instead of releasing the original (confidential) data set, the companies may provide only several statistical values of their data, such that a completely anonymous data set is automatically generated with similar characteristics.

From a practical point of view, such a method is beneficial to several parties. First of all, academic researchers can work on recent and realistic information. For instance, for studies on software development effort (henceforth, referred simply as effort) estimation, the proposed method is expected to be very helpful in assessment of stability, which inherently requires numerous industry data sets [77]. Thereby, validity and reliability of new effort estimation methods can better be assured. The proposed method is potentially useful also for the practitioners. Namely, companies may want to compare their software development performance metrics (such as productivity and defect density) with other companies. The proposed method enables comparison of performance through artificially generated data sets replicating statistical features of authentic data. We expect the proposed approach to encourage the companies to share the statistics of their data, once the researchers release their findings, which potentially involve beneficial information ready to be transferred to industry applications.

The principals of the proposed method are as follows. Regarding a real industry software project data set, we consider certain (authentic) variables (e.g. software metrics) and measure their statistics as well as pairwise correlation relations. Next, to generate synthetic variables, we use the Box-Muller transform [13] and obtain a set of normally distributed random numbers. Subsequently, we apply exponential transformation on those and transfigure the resulting values such that their (value) distribution emulates that of the authentic variables. After obtaining all such synthetic variables, number reordering is applied to achieve similar pairwise correlation relation to that within the authentic variables. In this respect, we assume that certain statistical information regarding an industrial data set (i.e. mean, standard deviation and correlation coefficient matrix) are not confidential and we expect to receive them as inputs to generate a mimic data set. Nevertheless, this assumption does not jeopardize confidentiality of the -input- data set, since the proposed method prevents identification of *any specific project* in this set, as it is a common requirement in data anonymization studies.

This chapter extends our previous work [28] with extensive empirical evaluation carried out on 8 industry data sets. In addition, we confirm the predictive ability of mimic data sets by illustrating the efficacy of synthetic variables for the particular purpose of effort estimation.

This chapter is organized as follows. We elaborate on the background and relevant studies in Section 2.2. Section 2.3 first gives an outline of the proposed method and then details each stage, whereas Section 2.4 provides a demonstration of the procedure on a commonly used data set. Subsequently, Section 2.5 considers effort estimation as one of the potential many application domains of mimic data; and evaluates and compares estimation performance obtained from authentic data and mimic data. Section 2.6 provides a discussion on the experimental validation, whereas Section 2.7 concludes the chapter summarizing our main results, contributions and future work.

2.2 Background and related work

Some of the most popular industry data sets employed in empirical software engineering studies such as Desharnais [20], Coc81dem [6], Kemerer [40], and Albrecht [3], are available at [70]. These data sets are all recorded in the 1980's. In this respect, the development environments and processes greatly differ from modern software development. In addition, the sample size is often very small, e.g. Kemerer has only 15 projects, whereas Albrecht has 24 projects. Surprisingly, although these data sets are old and small, they are still actively used in various recent studies appearing in top journals (e.g. [77, 2, 49]) due to the lack of more recent industry data sets.

On the other hand, there exist also a few self-contained studies based on recent software development data. However, they only report the analysis results and do not disclose any of the data itself. For example, the white paper on software development data in 2018-2019 [101] provides various analysis results of 4564 software development projects carried out by 34 Japanese software development companies. But it does not release the data set.

To mitigate the problems due to use of outdated or small sets, it is proposed to apply *anonymization* on recent software data sets. Data anonymization aims removing any identifying information from the original data such that the source or its private characteristics cannot be determined. Conventional data anonymizing methods for software engineering data employ *data mutation* techniques to gain data privacy [79, 80]. Since data mutation keeps the one-to-one mapping of data points between the anonymized data set and the original data set, threats of breaking the anonymity cannot be perfectly prevented. Moreover, since strong data mutation yields change of data characteristics, balancing privacy and utility is a big challenge [80].

In [79], Peters and Menzies proposed a data anonymization method called MORPH to solve privacy issues in software development organizations. They target defect prediction research and try to anonymize the defect data set that consists of various software metrics measured for each source file of a software product. They use data mutation techniques, which add small amount of changes to each value to make it difficult to identify a specific source file in a data set. They further propose a method called CLIFF, which allows to eliminate some data points that are not necessary for the defect prediction. Combining CLIFF with MORPH, they try to balance privacy and utility of defect data sets [80].

Since their approach is specifically proposed for a binary classification problem (i.e. distinguishing defect-prone and not-defect-prone files in a defect data set), it cannot be applied to general purpose data sets such as software project data sets as we target in this chapter.

2.3 Proposed method

Software project data sets typically involve various variables including software size metrics (e.g. function point, source lines of code), as well as project duration, and effort. As an example for a software project data set, Table 2.1 depicts an excerpt from the Desharnais data set[20], which is one of the commonly used software project data sets in effort estimation studies.

As it can clearly be seen in Table 2.1, the variables can be measured at varying scales. Namely, for the specific case of [70], *language* emerges as a nominal variable, whereas *team experience* and *project manager experience* are ordinal. On the other hand, quantitative variables involve such ratio scale variables as *function point*, *effort* and *duration*. Many software companies record such data sets consisting of project features similar to those listed in Table 2.1 [1]. Henceforth, we refer to such an authentic confidential data set as "source

Table 2.1: An example of software project data set (excerpt from Desharnais data set [20].)

TeamExp (years)	ManagerExp (years)	Duration (months)	Transactions	Entities	PointsAdjust	Lang2	Lang3	Effort (person-hours)
1	1	5	78	99	177	0	0	2520
4	7	13	69	74	143	0	0	1603
1	3	8	194	97	291	1	0	3626
1	3	10	42	31	73	1	0	1267
0	4	6	97	42	139	0	1	546
4	5	26	482	227	709	1	0	9100

[1]In this chapter, we assume that there is no missing value in a data set.

data set" or as simply "source data", whereas the synthetic data emulate the characteristics of the source data is referred as "mimic data set" or "mimic data".

Section 2.3.1 introduces the outline of the proposed method in a nutshell, whereas the details of the procedure are elaborated on in Sections 2.3.2, 2.3.3 and 2.3.4.

2.3.1 Outline

For generating mimic variables, we exploit the fact that probability distributions of the quantitaive variables in software project data sets roughly follow a log-normal distribution [48]. From this viewpoint, we approximate their (value) distribution with a log-normal distribution.

We initially generate a set of variables based on log-normal distribution assumption and obtain any number n of artificial values with a similar distribution to that of the corresponding values in source data. At this point, we are clearly required to follow a different approach for variables of different scale (i.e. nominal, ordinal, ratio and interval scale), which will be distinguished in Section 2.3.2 and Section 2.3.3.

Subsequent to obtaining a set of randomly generated values, we compute the correlation between each pair of variables and build the correlation matrix. Comparing it to the correlation matrix relating the source data, we try to emulate similar pairwise relations. To that end, we keep swapping the positions of the variables such that the correlation matrix of the mimic data resembles enough to that of the source data.

In relation to the above procedure, we would like to point out to two particular advantages. First of all, the proposed method supports any number of data points to generate. For example, we can generate mimic data with a sample size of $n = 1000$ from a source data of much smaller sample size, e.g. $n = 30$. Relying on this property, our second advantage is recognized as the lack of one-to-one correspondence between the projects of the authentic data set and the values in the mimic data set. Thanks to this, data privacy and confidentiality are suggested to be effectively protected, even if the mimic data set is made open.

2.3.2 Generation of ratio scale variables

Suppose that a certain quantitative variable in the source data set has a mean of m and a standard deviation of σ^2. Further, assume that after log-transforming it, the resulting distribution has a mean and standard deviation of $\widehat{m}$ and $\widehat{\sigma}^2$, respectively. Clearly, there exist the following relations between these pairs (see [97] for a detailed explanation),

$$m = \ln \left(\frac{\widehat{m}}{\sqrt{\frac{\widehat{\sigma}^2}{\widehat{m}^2} + 1}} \right), \tag{2.1}$$

$$\sigma^2 = \ln \left(\frac{\widehat{\sigma}^2}{\widehat{m}^2} + 1 \right).$$

For generating mimic values emulating the charateristics of the log-transformed quantitative (i.e. ratio scale and interval scale) variables, this chapter employs the Box-Muller transform [13], which is a pseudo-random number sampling method.

Essentially, Box-Muller transform generates a pair of independent and normally distributed random numbers from a given set of uniformly distributed random numbers. Let R_1 and R_2 be two independent random variables drawn from a uniform distribution in the interval (0,1)[2]. Box-Muller method, transforms these values into independent random variables N_1 and N_2 as follows,

$$N_1 = \sigma^* \sqrt{-2 \log R_1} \cos 2\pi R_2 + m^*, \tag{2.2}$$

$$N_2 = \sigma^* \sqrt{-2 \log R_1} \sin 2\pi R_2 + m^*,$$

where m^* and σ^* denote the desired mean and standard deviation of N_1 and N_2. Note that this procedure can be used also for interval scale variables.

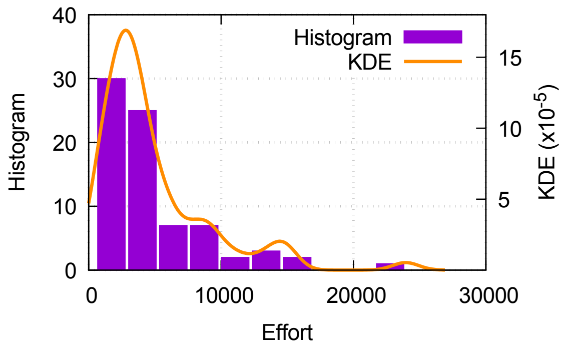
In our specific application, since we target generating values emulating the log-transformed distribution, we use $m^*= \widehat{m}$ and $\sigma^*= \widehat{\sigma}$. Moreover, this chapter utilizes only N_1.

As mentioned in Section 2.3.1, we assume that quantitative variables follow a log-normal distribution. Therefore, we apply exponential transformation on N_1 and obtain the mimicking values of the variables.

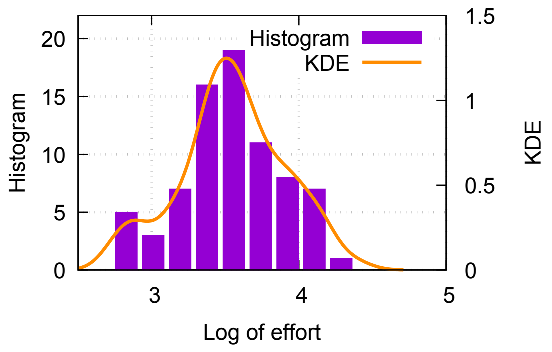
Figure 2.1 gives a demonstrating example of this process depicted on the effort variable concerning Desharnais data set. The distribution is illustrated both in terms of a histogram and a kernel density estimate (KDE). Figure 2.1-(a) relates the source data, whereas Figure 2.1-(b) illustrates its log-transform[3]. We use the mean value m and standard deviation

[2] R_1 and R_2 can easily be generated in many programming languages, e.g. by using `rand()` function in C programming language.

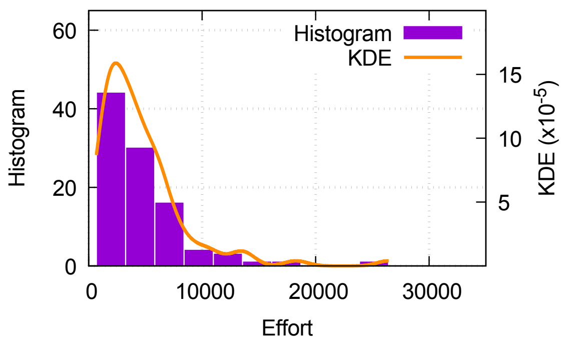
[3] Figure 2.1-(b) confirms that log-transformed values roughly follow the normal distribution.



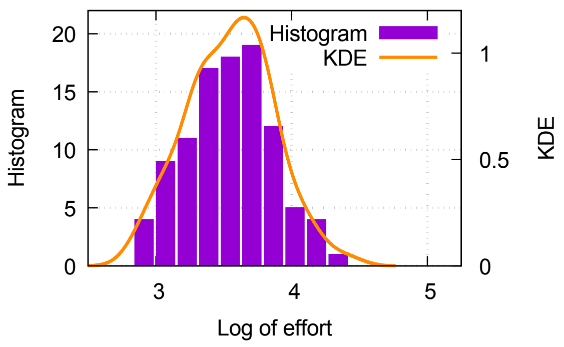
(a)



(b)



(c)



(d)

Figure 2.1: Histogram and kernel density estimates regarding (a) raw and (b) log-transformed values of effort of Desharnais data set; and (c) mimic data after exponential transformation and (d) the mimic data.

σ of the authentic effort values to obtain the desired statistics of the log-transformed distribution using Equation 2.1 and generate the mimic data using Equation 2.2. Figure 2.1-(c) shows the outcome of this operation in terms of the histogram of the mimic variables and relating KDE. Finally, Figure 2.1-(d) shows the result of exponential transformation applied on the mimic variables. Although values in Figure 2.1-(d) are derived artificially generated, we see that the distributions are well in line with those of the source data presented in Figure 2.1-(b).

For realizing the above procedure, a company that owns a (confidential) software development data set, needs to provide only m and σ, which are directly computed from the source data.

2.3.3 Generation of ordinal and nominal scale variables

For each ordinal scale or nominal scale variable in the source data, we generate a set of artificial values so that the percentage of cases in each bin is same as the source data.

For instance, consider that we have an ordinal scale variable as "requirement clarity", which has four ranks (or bins) as "1. very clear", "2. clear", "3. unclear", "4. very unclear".

Suppose that the percentage of values belonging to each bin are 20% for "1. very clear", 25% for "2. clear", 40% for "3. unclear" and 15% for "5. very unclear", respectively. In order to generate mimic data, which represents the characteristics of the authentic distribution, we simply generate an artificial mimic sample, whose percentage of cases corresponding to each bin is same as that of the source data.

2.3.4 Mimicking pairwise relationships of the variables

Between every pair of variables in the source data, there exist a certain relationship, which we opt to capture via a correlation matrix χ. Specifically, we use Spearman's rank correlation coefficient instead of the common practice based on Pearson correlation coefficient (see Algorithm 1[4]). This choice is due to the existence of outliers in the source data.

Algorithm 1: Computation of correlation matrix based on Spearman's rank correlation coefficient.

```
Input: Values of project variables $\overrightarrow{V}_i$, $\forall i \in [1, T]$
Output: Correlation matrix $\chi$
1 for $i \leftarrow 1\ :\ T$ do
2     Set $F$ to size of $\overrightarrow{V}_i$
3     Set $\overrightarrow{S}_i$ to ranked array of $\overrightarrow{V}_i$
4     for $i \leftarrow 1\ :\ F$ do
5         for $j \leftarrow 1\ :\ F$ do
6             $\chi(i,j) = \frac{\texttt{cov}(S_i,S_j)}{\sigma_{Si}\sigma_{Sj}}$
```

[4]In Algorithm 1, cov stands for covariance, σ stands for standard deviation, and T stands for number of project variables.

Algorithm 2: Mimicking pairwise relations.

```
   Input: Values of mimic variables $\overrightarrow{V}'_i$ $\forall i \in [1,T]$
   Correlation matrix of source data $\chi$                    // See Alg. 1
   Output: Reordered values of mimic variables $\overrightarrow{V}'_i$, $\forall i \in [1,T]$
1  Set $S'$ to size of any array $\overrightarrow{V}'_i$
2  Set $\varepsilon_0 = \infty$                        // Previous value of $\varepsilon$
3  do
       /* Reorder by swapping arbitrary values */
4      Get a pair of arbitrary indices $1 < p, q < S'$ , $i \neq j$
5      $\overrightarrow{V}'_i(p) \leftrightarrow \overrightarrow{V}'_i(q)$                                  // Swap
6      Get $\chi'$ relating reordered mimic data                  // See Alg. 1
       /* Similarity of correlation matrices $\varepsilon$ is expressed in terms of sum
          of squared differences.  */
7      Set $\varepsilon = \sum_{i,j=1}^{T} (\chi(i,j) - \chi'(i,j))^2$
8      if $\varepsilon < \varepsilon_0$ then                     // There is improvement
9
10         $\varepsilon_0 = \varepsilon$
11     else                                     // There is no improvement
12
13         $\overrightarrow{V}'_i(p) \leftrightarrow \overrightarrow{V}'_i(q)$                       // Swap back
14 while $\varepsilon$  converging
```

Based on the correlation matrix, we emulate a similar pairwise relation between every possible mimic variable pair to that of the authentic variable pairs. To that end, we apply number reordering to the array of the mimic data. Namely, we swap random values, which obviously does not have any effect on the distribution of that variable. Specifically, we employ the procedure presented in Algorithm 2, which evaluates similarity of the correlation matrices χ and χ' concerning source and mimic data in terms of sum of squared differences ε[5].

Subsequently, we make mimic data visually more similar to source data by rounding-off to

[5]Here, convergence is judged in terms of the number of successive iterations which do not lead to an improvement.

a suitable precision. Namely, mimicking values of the quantitative variables are generated from random numbers, and thus their significant figures are different from those in the source data. Therefore, each mimicking value should be rounded off to an appropriate precision according to the significant figure in the source data. For instance, Function Point variable is an integer in the source data, and as such, it is rounded off to integer.

2.4 Case study on the Desharnais data set

In order to demonstrate the operation of the procedure introduced in Section 2.3, we present a case study carried out on the Desharnais data set[20], which is one of the most frequently used data sets in effort estimation research [48].

Desharnais data set contains 77 projects without missing values. Here, we generate mimic data with a sample size of $n = 100$. Besides, mimic data composes of 5 quantitative variables as Duration, Transactions, Entities, PointsAdjust, and Effort; and 3 qualitative variables as TeamExp, ManagerExp, and Lang. Here, TeamExp and ManagerExp are ordinal scale variables, where TeamExp ranges from 0 to 4, and the ManagerExp ranges from 0 to 7. In addition, the variable Lang is divided into two binary variables as Lang2 and Lang3.

2.4.1 Characteristics of quantitative variables

For an arbitrary quantitative (i.e. ratio or interval scale) variable r, we denote the mean value, standard deviation, minimum and maximum with m, σ, $r_{\min}$ and $r_{\max}$, respectively.

Table 2.2 Statistics of source data.

	m	σ	$r_{\min}$	$r_{\max}$
Duration	11.30	6.74	1	36
Transactions	177.47	145.13	9	886
Entities	120.55	85.55	7	387
PointsAdjust	298.01	181.08	73	1127
Effort	4833.91	4160.90	546	23940

Table 2.3 Statistics of mimic data.

	$\widehat{m}$	$\widehat{v}$	$\widehat{r}_{\min}$	$\widehat{r}_{\max}$
Duration	11.571	7.172	3	42
Transactions	180.078	139.485	39	822
Entities	123.208	91.018	29	534
PointsAdjust	300.299	168.783	99	986
Effort	4913.26	4246.176	893	25365

Table 2.4 Relative difference of statistics.

	Δm	$\Delta \sigma$	$\Delta r_{\min}$	$\Delta r_{\max}$
Duration	0.02	0.03	0.5	0.14
Transactions	0.03	0.16	0.44	0.37
Entities	0.02	0.07	0.41	0.04
PointsAdjust	0.03	0.14	0.32	0.34
Effort	0.04	0.13	0.39	0.30

Table 2.2 illustrates these values for the source data set, while Table 2.3 presents similar values relating mimic data, which are represented with $\widetilde{m}$, $\widetilde{\sigma}$, $\widetilde{r}_{\min}$ and $\widetilde{r}_{\max}$.

In addition, in Table 2.4 we present the relative differences of the statistics given in Tables 2.2 and 2.3. For instance for mean value, relative difference is defined as,

$$\Delta m = \frac{|m - \widetilde{m}|}{\max(m,\ \widetilde{m})}. \tag{2.3}$$

From these results, we see that the absolute difference of mean value, standard deviation and minimum value between two data sets are very small, which indicates effectiveness of the proposed method. Note that, the relative difference values relating the minimum are higher than those relating the maximum. However, it is sufficient to check the values in Tables 2.2 and 2.3 to realize that they are inflated due to the inherently small values of $r_{\min}$ and the distributions still attain very similar minimums. On the other hand, the maximum values turn out to be not very similar, principally because source data set contains outliers.

2.4.2 Characteristics of the correlation matrix

Parts of the correlation matrices regarding source data and mimic data are shown in Table 2.5 and Table 2.6. In addition, Table 2.7 presents the absolute value of element-wise differences of Table 2.5 and Table 2.6. From Table 2.7, it can be observed that the maximum difference is attained for a particular pair of variables, namely of Lang2 and Lang3. This maximum difference of 0.023 is considered to be sufficiently small. Therefore, the relationship between any two variables is regarded to be effectively reproduced.

In addition, Figure 2.2 shows the convergence of the sum of squared differences ε of rank correlation coefficients for increasing number of updates (i.e. successful swapping of variables). As shown in the figure, ε becomes very close to zero for growing number of iterations (e.g. 0.20495 at 1000 iterations and 0.000216 at 10000 iterations).

Table 2.5 Correlation matrix for the source data.

	TeamExp	ManagerExp	Duration	Transactions	Entities	PointsAdjust	Lang2	Lang3	Effort
TeamExp	1.000								
ManagerExp	0.388	1.000							
Duration	0.365	0.233	1.000						
Transactions	0.088	0.109	0.382	1.000					
Entities	0.319	0.170	0.533	0.265	1.000				
PointsAdjust	0.266	0.189	0.592	0.744	0.778	1.000			
Lang2	-0.072	0.157	0.147	-0.129	0.045	-0.039	1.000		
Lang3	-0.078	0.180	-0.106	0.248	-0.120	0.077	-0.247	1.000	
Effort	0.252	0.086	0.572	0.467	0.647	0.688	0.022	-0.428	1.000

Table 2.6 Correlation matrix for the mimic data.

	TeamExp	ManagerExp	Duration	Transactions	Entities	PointsAdjust	Lang2	Lang3	Effort
TeamExp	1.000								
ManagerExp	0.389	1.000							
Duration	0.365	0.235	1.000						
Transactions	0.088	0.109	0.381	1.000					
Entities	0.319	0.170	0.532	0.265	1.000				
PointsAdjust	0.266	0.189	0.591	0.742	0.776	1.000			
Lang2	-0.071	0.165	0.145	-0.128	0.045	-0.039	1.000		
Lang3	-0.067	0.187	-0.106	0.248	-0.120	0.077	-0.224	1.000	
Effort	0.252	0.086	0.572	0.466	0.647	0.690	0.022	-0.427	1.000

Table 2.7 Absolute value of difference of correlation matrices.

	TeamExp	ManagerExp	Duration	Transactions	Entities	PointsAdjust	Lang2	Lang3	Effort
TeamExp	0								
ManagerExp	0.001	0							
Duration	0.002	0	0						
Transactions	0	0	0.001	0					
Entities	0	0	0.001	0	0				
PointsAdjust	0	0	0.001	0.002	0.002	0			
Lang2	0.001	0.008	0.002	0.001	0	0	0		
Lang3	0.011	0.007	0	0	0	0	**0.023**	0	
Effort	0	0	0	0.001	0	0.002	0	0.001	0

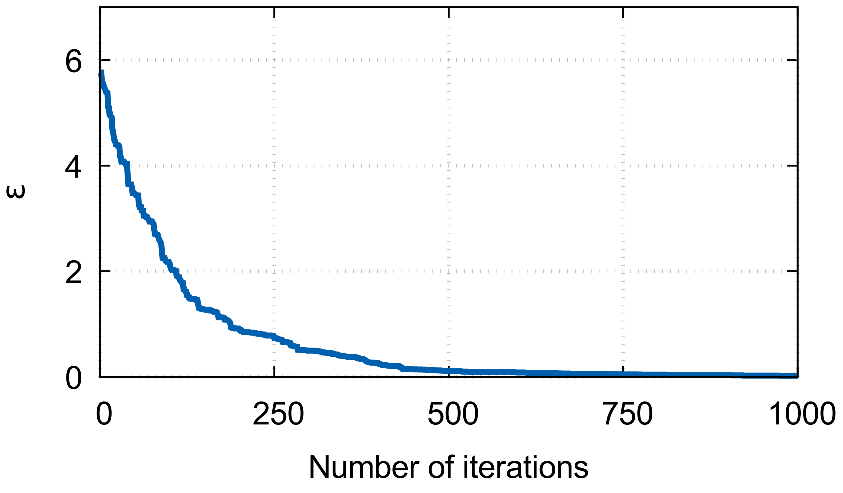


Figure 2.2 Convergence of sum of squared differences ε of rank correlation coefficients.

2.5 Evaluation based on effort estimation performance

To evaluate utility of the mimic data set, we consider effort estimation as a representative application domain for employing mimic data. Namely, we consider a source data set with a set of variables composed of effort and several others. From this source data set, we generate mimic data regarding the variables other than effort. Based on this artificially generated set, we carry out effort estimation. On the other hand, we estimate effort based on the authentic variables in the source data set. We compare both estimations to the true values of effort. Effort estimation performance obtained using the authentic variables (i.e. source data set) is

Table 2.8 Reference data sets employed in the experiments.

Dataset	Number of categorical variables	Number of continuous variables	Number of projects
Albrecht [3]	0	7	24
China [70]	1	11	499
Coc81dem [6]	14	4	63
Desharnais [20]	4	5	77
Kemerer [40]	2	5	15
Maxwell [64]	22	4	62
Miyazaki94 [72]	0	8	48
Nasa93 [70]	24	2	93

considered as benchmark performance. We compare this to the performance rates obtained using mimic data to investigate whether estimation performance based on mimic data is similar to the performance of the benchmark estimation obtained using authentic data.

To that end, in our experiments we employ as source data 8 data sets (henceforth, noted as reference data sets) introduced in Table 2.8. All data sets used in this study are publicly available [70].

2.5.1 Effort estimation model

As mentioned above, we consider effort estimation to be one of the many possible areas of deployment of mimic data. On this basis, we carry out the effort estimation method described below.

Specifically, this study performs effort estimation (in person-months or person-hours) based on linear regression modeling. Generally speaking, a linear regression model is described as follows:

$$\widehat{Y} = \sum_{j=1}^{n} k_j N_j + C \tag{2.4}$$

$\widehat{Y}$	: Estimated value of the objective variable
N_j	: Predictor variables
k_j	: Partial regression coefficients
C	: A constant

For our specific case, the linear regression model employs effort as the objective variable and remaining variables as (potential) predictor variables. As mentioned in Section 2.3.1 and illustrated in Figure 2.1, logarithmic transformation of project variables yields a more similar distribution to normal distribution. Although linear regression does not explicitly require neither the objective variable or the predictor variables to come from a normal distribution[6], as pointed out by Kitchenham and Mendes [48], their logarithmic transformation is empirically shown to help improving estimation results obtained by linear regression. Thus, it is a commonly used preprocessing operation in linear regression model construction, which also we opt to follow in this study.

In this respect, as a preprocessing operation, logarithmic transformation is applied on both the objective variable (i.e. effort) and the predictor variables (e.g. function point, duration etc) prior to model construction. In addition, for variables containing 0, an offset value (such as 0.5 or 1) is added before the transformation.

Thereby, the estimation model boils down to a log-log regression, which is expressed simply as follows:

$$\log \widehat{Y} = \sum_{j=1}^{n} k_j \log N_j + C. \tag{2.5}$$

It follows that, by applying exponential transformation on Equation 2.5, the estimated value $\widehat{Y}$ can be obtained as:

$$\widehat{Y} = \exp(C) \prod_{j=1}^{n} {N_j}^{k_j}. \tag{2.6}$$

2.5.2 Selection of predictor variables

As explained in Section 2.5.1, effort is the objective variable and remaining variables are potential predictor variables. In other words, we do not employ all available variables of a data set in effort estimation and instead eliminate any irrelevant or useless variables, which is one of the most crucial factors acting on estimation performance.

[6]Linear regression assumes normality of residual errors.

Algorithm 3: Selection of predictor variables.

```
Input: Set of all project variables $V = \{V_1, \ldots, V_T\}$
Output: Set of predictor variables $N, k, C$
1  Choose an arbitrary $i \in [1, T]$
2  Set $N = V_i$
3  while True do
       /* Insertion of a variable */
4      Choose an arbitrary $j \in [1, T]$, $V_j \notin N$
5      $N' = N \cup V_j$
       /* Removal of a variable */
6      Choose an arbitrary $k \in [1, T]$, $V_k \in N$
7      $N'' = N \setminus V_k$
       /* Difference in AIC */
8      $\delta' = AIC(N') - AIC(N)$
9      $\delta'' = AIC(N'') - AIC(N)$
10     if $\delta' < 0 \parallel \delta'' < 0$ then                // There is improvement
11
12         if $\delta' < \delta''$ then
13             $N = N'$
14         else
15             $N = N''$
16     else                                       // There is no improvement
17
18         break
```

To that end, this study uses an iterative variable selection procedure based on Akaike's Information Criterion (AIC) [1] as depicted in Algorithm 3. Namely, we first build a simple model with a single predictor variable. At each iteration, we modify the model (i) by inserting an additional predictor variable and (ii) by removing a single predictor variable. This modification yields one extended model and one simplified model as compared to original one. Comparing the three AIC values concerning the original, extended and simplified models, we choose the best performing set of variables and update the model. We pursue this procedure until no extension or simplification brings any performance enhancement (in terms of AIC).

2.5.3 Evaluation procedure

For each data set presented in Table 2.8, we conducted 10 repetitions of the 3-fold cross validation procedure illustrated in Figure 2.3. We expect this series of experiments to mitigate the effect introduced by the random splitting (of the source data into test and fit data) on estimation results and provide an insight to the stability of performance.

Specifically, we randomly split a source data set into 3 subsets. Then, we conduct three sets of model construction and model evaluation, each of which employs two subsets in model construction. In each case, the remaining subset is used in evaluation. We then construct an effort model for each original (source) fit subset and mimic data derived from fit subset.

Evaluation of both models are done using only the test subset. In other words, we do not

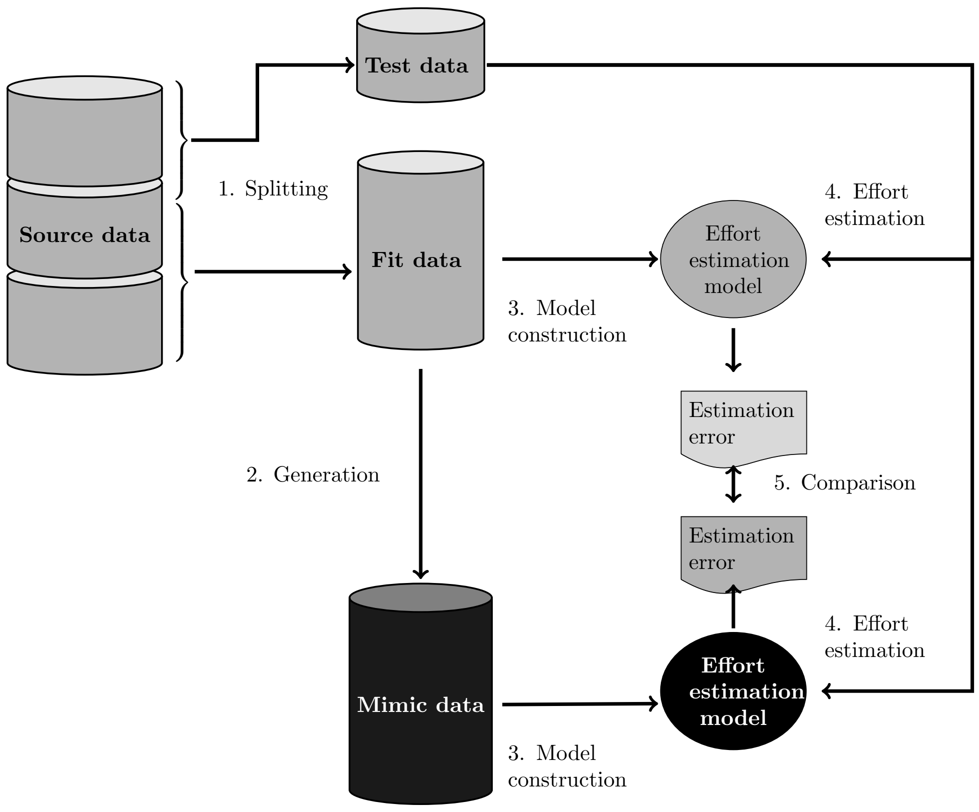


Figure 2.3 The 3-fold cross validation scheme.

produce mimic data for the test subset, since this experiment aims to evaluate how mimic data performs in effort estimation of real source data (i.e. not the mimic data).

To evaluate the effectiveness of our method, we apply this procedure on each reference data set and evaluate the performance using the metrics defined in Section 2.5.4.

2.5.4 Evaluation criteria

As $\widehat{Y}$ denotes estimated value (see Equation 2.6) and Y denotes the true value of effort, let absolute error (AE), magnitude of relative error (MRE), magnitude of error relative to the estimate (MER), and balanced relative error (BRE) be defined as,

$$\begin{aligned}\mathbf{AE} &= |Y - \widehat{Y}|,\\ \mathbf{MRE} &= \frac{|Y - \widehat{Y}|}{Y},\\ \mathbf{MER} &= \frac{|Y - \widehat{Y}|}{\widehat{Y}},\\ \mathbf{BRE} &= \frac{|Y - \widehat{Y}|}{\max(Y, \widehat{Y})}\end{aligned} \tag{2.7}$$

As for evaluation criteria, we use the above values, namely the relative difference of means between the source-based and mimic-based effort estimation; the difference of means between the source-based and mimic-based effort estimation relative to the mean of source data set, relative to the mean of mimic data set and relative the larger one of mean of source and mimic data sets. Let $\boldsymbol{m}(\cdot)$ and $\boldsymbol{\sigma}(\cdot)$ stand for functions returning mean and standard deviation, respectively. Consider relative difference is defined as in Equation 2.3. Then, our evaluation criteria are expressed simply as $\Delta\boldsymbol{m}(\mathbf{AE})$, $\Delta\boldsymbol{m}(\mathbf{MRE})$, $\Delta\boldsymbol{m}(\mathbf{MER})$, and $\Delta\boldsymbol{m}(\mathbf{BRE})$.

For the two effort estimation procedures (one being based on source data and the other being based on mimic data), if the above-mentioned mean values are similar, the proposed method is concluded to be effective in effort estimation.

2.5.5 Evaluation results

Table 2.9 presents estimation performance based on the criteria presented in Section 2.5.4. Obviously, most of the values are quite low, indicating that similar **AE**, **MRE**, **MER** and **BRE** values are achieved by effort estimation using source data and mimic data. However,

Table 2.9 Evaluation results.

Dataset	$\Delta\boldsymbol{m}(\mathbf{AE})$	$\Delta\boldsymbol{m}(\mathbf{MRE})$	$\Delta\boldsymbol{m}(\mathbf{MER})$	$\Delta\boldsymbol{m}(\mathbf{BRE})$
Albrecht	0.63	0.02	0.30	0.01
China	0.02	0.01	0.03	0.01
Coc81-dem	0.10	0.05	0.17	0.08
Desharnais	0.02	0.05	0.02	0.02
Kemerer	0.33	0.53	0.44	0.18
Maxwell	0.04	0.03	0.08	0.02
miyazaki94	0.04	0.11	0.09	0.07
nasa93	0.04	0.02	0.30	0.08

there are a few individual cases, which stand out. For instance, the values of $\Delta\boldsymbol{m}(\mathbf{AE})$ relating Kemerer and Albrecht data sets are quite high ($0.33 \sim 0.63$). This is primarily due to limited number of samples in these data sets (see Table 2.8). This disadvantage of Kemerer and Albrecht data sets can be observed also in terms of the other performance metrics. Namely, for Kemerer, **MRE**, **MER** and **BRE** are all significantly higher compared to other data sets, whereas for Albrecht **MER** sticks out as usually large.

In addition, we notice that in general **MER** is larger than **MRE**. This is possibly due to the fact that it is hard to mimic the projects, whose variables belong to the tails of the distributions. Therefore, since the proposed method has a tendency to produce mimic data resembling the bulk of the distribution, the estimated value $\widehat{Y}$ is likely to be lower, which in turn increases **MER** (see Equation 2.7). On the other hand, **BRE** introduces better (i.e. lower) rates, since it considers a larger value in the denominator (see Equation 2.7).

2.6 Threats to validity

We provide a discussion on the validity of the proposed method in terms of three commonly adopted experimental validation approaches, i.e. internal validity, external validity and construct validity.

Internal validity refers to the extent by which the observed effect is a consequence of the presumed cause. In our case, internal validity questions whether or not different conclusions can be drawn with regard to the different settings in the experiment. To ensure the internal validity, we conducted 10 repetitions in the validation process to produce the stable result. However, there are several possible issues of internal validity in this study. One issue is the single sampling method (2-fold cross validation) we used. Our important future work is to employ other method such as leave-one-out cross validation to increase the validity of the result. Second issue is the single modeling method we used. We chose log-log regression model with stepwise variable selection, which is one of the most commonly used modeling techniques in software effort estimation. It is our future work to employ other modeling methods.

External validity refers to the generalization of the results. In this study, we address external validity by using 8 reference data sets with diverse characteristics. Namely, they vary in size (i.e. number of projects), and project variables, as well as origin (i.e. recording organization) and recording period. We believe that a diversity of data sets being can produce generalized findings.

Construct validity refers to the relevance and capability of the observations and measurements in evaluating the posed hypothesis. In this study, we address construct validity by using both absolute error and relative error in the evaluating the effort estimation performance. However, since there are other measures of estimation error such as balanced relative error, it is our future work to employ such measures to increase the validity of our work.

2.7 Conclusions

This study proposes a method for building a mimic data set replicating the statistical properties of a (confidential) source data set. Software development companies, which cannot release data sets, can provide only a couple of data statistics (i.e. mean and standard deviation), which enable generating a mimic data set of any desired size with similar characteristics to the authentic data, by complying to legal requirements on privacy as well as respecting meeting the demands of the stakeholders.

Based on our case study, we demonstrated that the mimic data set follows the characteristics of the source data set with considerable similarity. As for a potential application domain for utilizing mimic data, we consider effort estimation. By experimentally evaluating the effort estimation performance of mimic data sets derived from 8 different reference data sets, which are commonly used in effort estimation studies, we proved that performance rates achieved by mimic data are quite similar to those achieved by source data, provided that the data set has a sufficient number of samples (i.e. projects).

Based on these results, we claim that data anonymization is achieved effectively and the proposed method is a superior alternative to data mutation. Specifically, since all data points are artificially produced from randomly produced normal distribution values without referring to data points in the source data set, and the number of data points in the mimic data set can be set to any desired value, one cannot find one-to-one data mapping between source data and mimic data. We expect these results to be encouraging reasons for the companies to release statistics of their data for generation of mimic data sets; and both the research community and the industrial partners to profit from the outcomes of this study.

As future work, we consider applying various other data analysis techniques such as clustering and association rule mining for mimic data to evaluate the utility of the proposed method. In addition, we will try to improve our method by mimicking other characteristics of source data (in addition to mean and standard deviation), such as outliers, skewness and kurtosis.

Chapter 3

Improvement and Evaluation of Data Consistency Metric CIL

3.1 Introduction and motivation

In early stages of a software development project, various target values, such as development effort, software productivity, defect density, etc. need to be estimated, typically by referring to the data from other past projects using machine learning techniques [4, 15, 92]. However, if the quality of such data is low, these estimation techniques may not work efficiently or may behave in unexpected ways [56].

Although the benefits of assessment of data quality are self-evident, according to a systematic review by Liebchen *et al.* , only 23 out of hundreds of articles explicitly addressed this issue[58]. In that respect, Liebchen *et al.* emphasize that researchers should pay more attention to the quality of data, before deploying it.

According to the taxonomy proposed by Bosu *et al.* [14, 15], data quality challenges in empirical software engineering (ESE) include specifically outliers [60, 75], noise [24, 52, 57], data incompleteness [11, 57, 83], data inconsistency [17, 56, 31] and redundancy [23].

To the best of our knowledge, Case Inconsistency Level (CIL) is the only data inconsistency metric proposed to date [78]. According to [78], inconsistency is characterized by project cases with *conflicting feature values.* Table 3.1 illustrates such a conflict on an excerpt from the China data set [70]. Specifically, each row of Table 3.1 represents a software project and each column represents a project feature [70, 45]. When we look closely at projects ID 2

Table 3.1 An example of software project data set (excerpt from China data set [70].

ID	AFP (FP)	Input	Output	Enquiry	File	Interface	Duration	Effort (person-hours)
1	919	458	236	56	58	76	6	4815
2	66	27	12	6	14	0	2	246
3	54	36	0	16	0	0	1	686
4	151	41	21	29	41	19	3	391
5	66	27	14	6	17	0	2	1817
6	70	24	0	3	42	5	3	540
7	754	301	130	204	90	0	8	7443

and ID 5, we see that they appear to have very similar characteristics, but the development effort of project ID 5 is more than five times greater than that of project ID 2. CIL evaluates the quality of a data set based on the number of such inconsistent pairs.

In this study, we first conduct a follow-up evaluation of CIL using a large sample set, and show that CIL is not effective in assessing the quality of certain data sets. Based on our analysis of ineffective cases, we propose an improved metric called Similar Case Inconsistency Level (SCIL).

Incorporating SCIL with various distance metrics and pre-processing methods, we empirically determine an efficient execution mode[1]. Applying it on 18 sets derived from six large project data sets from the SeaCraft repository of ESE data [70][2], we distinguish between consistent and inconsistent data sets. Finally, we show that effort/productivity estimation models built from data sets identified as consistent by SCIL achieve indeed a relatively high accuracy.

This chapter is organized as follows: We elaborate on the definition of CIL in Section 3.3 and discuss its issues in Section 3.4 through a follow-up evaluation. In Section 3.5, we propose a new improved metric SCIL and evaluate it in detail and compare it to CIL in Section 3.6. Section 3.7 provides a discussion on experimental validation. Finally, we conclude the article

[1]Here, by "efficient execution mode" we refer to the incorporation of SCIL with the best performing combination of distance metric and pre-processing methods (among those that we experimented with).

[2]SeaCraft stands for "Software Engineering Artifacts Can Really Assist Future Tasks", is an open access repository accumulating various software development project data sets provided to software engineering researchers and practitioners for analyzing/tackling real-life challenges.

and discuss future prospects in Section 3.8.

3.2 Background and related work

While there are many different definitions and views on data quality, the most widely accepted definition is based on "fitness for purpose" [14, 58, 81, 59]. Specifically, Mocnik *et al.* define "fitness for purpose" as the affordance of data to be interpreted and used in a context that renders a certain usage, that is, the purpose, possible [73].

Adopting the point of view, this study handles data quality in relation to a specific "purpose", namely "construction of effort/productivity estimation models". The reason for choosing this purpose is (i) the importance of effort/productivity estimation in software project planning and (ii) the large number of attempts in effort/productivity estimation from *empirical data* (i.e. based on actual historical data sets) [8, 29, 4, 82, 83, 91].

Bosu *et al.* [14] point out that data quality issues in ESE can be broken-down into three main classes as (i) accuracy, (ii) relevance and (iii) provenance. Specifically, accuracy refers to the correctness of the data[3], whereas relevance refers to the appropriateness of data for developing a model and provenance refers to the accessibility and trustworthiness of data[4].

The data quality assessment framework of this study excludes the use of non-relevant and inaccessible data, which can be associated also with the flaws in scientific procedure rather than issues with the data set. In that respect, we focus mainly on the accuracy aspect in determining data quality. Note also that from the point of view of our purpose (i.e. effort/productivity estimation), accuracy is particularly important[14], since high estimation accuracy is essential in efficient project planning and control.

According to [14], accuracy issue can further be broken-down into five sub-issues as outliers, noise, inconsistency, incompleteness and redundancy. We recognize that the aspects like outliers, noise, incompleteness, and redundancy are important factors that determine accuracy and, in turn, data quality. Nevertheless, we let them remain beyond the scope of this investigation and focus on data inconsistency in determining quality of a data set[5].

Liebchen *et al.* [57] state that inconsistent data is the one that cannot be easily explained.

[3]Namely, absence of noise.

[4]In other words, provenance is related to the possibility of experimental replication.

[5]In that respect, when we mention "quality of a data set" in the rest of text, we refer specifically "its data consistency".

More specifically, Bosu *et al.* [14] state that inconsistency means a lack of harmony between different parts or elements in a data set (i.e. instances conflicting within themselves or between each other).

According to the survey by Phannachitta *et al.*, at the time of their study there was no existing software metric that could directly quantify the level of consistency in ESE data sets [78]. This lack of procedure motivated Phannachitta *et al.* to develop a software metric for evaluating the inconsistency level of a data set. In that respect, they proposed a metric called Case Inconsistency Level (CIL)[78]. However, as explained in Sections 3.4 and 3.4.6, CIL has some serious problems, which this study identifies and offers a solution for.

3.3 Revisiting CIL

In this section, we will first define the variables and concepts essential for computing CIL. We will then explain the basic idea underlying CIL and provide its formal definition.

Let D be a data set of N software projects,

$$D = \{\mathbf{p}_i,\ 1 \leq i \leq N\}, \tag{3.1}$$

where $\mathbf{p}_i$ stands for the i^{th} project (or equivalently its feature vector).

Let the feature f_{m^*} represent the estimation target and consider that all other features, i.e. f_m such that $m \neq m^*$, are disposable for estimating its value. Henceforth, we refer to f_{m^*} as "estimation target variable" (or simply as "target variable") and f_m as "estimator variables".

Suppose that $\mathbf{P}_D$ stands for the set which contains all possible pairs of different projects belonging to the data set D,

$$\mathbf{P}_D = \{\mathbf{p}_{ij} \mid \mathbf{p}_i, \mathbf{p}_j \in D,\ i \leq j\}, \tag{3.2}$$

where $\mathbf{p}_{ij}$ is simply an unordered pair $(\mathbf{p}_i, \mathbf{p}_j)$[6].

3.3.1 Concepts essential in computing CIL

In this section, we will explain two concepts which are essential in computing CIL, namely the Interpolated Value Difference Metric (IVDM) and normalized rank of relative similarity.

[6]Note that $(\mathbf{p}_i, \mathbf{p}_j)$ and $(\mathbf{p}_j, \mathbf{p}_i)$ are essentially the same. In such, we opt for setting $i \leq j$ in Equation 3.2 so as to eliminate the redundancy in notation.

Interpolated Value Difference Metric

Let $Q(v)$ denote a function, which discretizes continuous input values v into C intervals with an equal width of δ. In addition, suppose that $\bar{Q}(v)$ is the mid-point of the discretization bin corresponding to input v.

Let $P'(Q(\mathbf{p}[f_{m^*}]) = c|\mathbf{p}[f_m])$ denote the conditional probability that the estimation target variable f_{m^*} is mapped to the discretization bin c given the value of the estimator variable f_m. This conditional probability can be expressed as in Equation 3.3.

$$\begin{aligned}P'\Big(Q(\mathbf{p}[f_{m^*}]) = c \mid \mathbf{p}[f_m]\Big) =& P\Big(Q(\mathbf{p}[f_{m^*}]) = c \mid Q(\mathbf{p}[f_m])\Big) + \\ & \frac{\mathbf{p}[f_m] - \bar{Q}(\mathbf{p}[f_m])}{\delta} \cdot \Bigg(P\Big(Q(\mathbf{p}[f_{m^*}]) = c \mid Q(\mathbf{p}[f_m])\Big) - \\ & P\Big(Q(\mathbf{p}[f_{m^*}]) = c \mid Q(\mathbf{p}[f_m]) + 1\Big)\Bigg)\end{aligned} \tag{3.3}$$

Interpolated Value Difference Metric (IVDM) is defined in terms of the difference in such conditional probabilities,

$$\begin{aligned}&\mathbf{IVDM}(\mathbf{p}_{ij}, f_{m^*}) = \\ &\sum_{f_m}\sum_{c=1}^{C}\Bigg(P'\Big(Q(\mathbf{p}_i[f_{m^*}] = c \mid \mathbf{p}_i[f_m])\Big) - \\ &\quad P'\Big(Q(\mathbf{p}_j[f_{m^*}] = c \mid \mathbf{p}_j[f_m])\Big)\Bigg)^2\end{aligned} \tag{3.4}$$

Normalized rank of relative similarity

The normalized rank of relative similarity is denoted with d_{NR} and computed based on the probabilistic similarity measure IVDM.

Let S be a set of real numbers and suppose that $\mathbf{rank}(s, S)$ returns the index of one of its elements s, when S is sorted in ascending order. Namely,

$$\mathbf{rank}(s, S) = \#\left(\{s'|s > s', s' \in S\}\right), \tag{3.5}$$

where $\#(\cdot)$ returns the number of elements of a set.

In order to compute d_{NR} relating to target variable f_{m^*}, firstly IVDM values concerning all pairs of projects in $\mathbf{P}_D$ (i.e. $\mathbf{IVDM}(\mathbf{P}_D)$) are computed. These values are then ranked using Equation 3.5 and normalized as in Equation 3.6,

$$d_{NR}(\mathbf{p}_{ij}, f_{m^*}) = \frac{\mathbf{rank}\left(\mathbf{IVDM}(\mathbf{p}_{ij}, f_{m^*}),\ \mathbf{IVDM}(\mathbf{P}_D, f_{m^*})\right)}{\#(\mathbf{P}_D) - 1}. \tag{3.6}$$

Note that here the rank value in the numerator is inherently between 0 and $\#(\mathbf{P}_D) - 1$, which means that $d_{NR} \in [0, 1]$[7].

The normalized rank of relative similarity is often considered to be a better distance function than the conventional Euclidean distance when used in predictive models [10, 16].

3.3.2 Definition of CIL

According to [78], a project pair $\mathbf{p}_{ij}$ is regarded to be inconsistent, if at least one of the following conditions is satisfied:

(R1) $\mathbf{p}_i$ and $\mathbf{p}_j$ are dissimilar in terms of the target variable f_{m^*}, although they are very similar in terms of the estimator variables f_m.

(R2) $\mathbf{p}_i$ and $\mathbf{p}_j$ are similar in terms of the target variable f_{m^*}, although they are dissimilar in terms of the estimator variables f_m.

In addition, a "consistent data set" is considered to be a data set *free of inconsistent pairs*, i.e. involving no cases of (R1) or (R2). On the other hand, an "inconsistent data set" is a data set with non-zero cases of (R1) or (R2), where the level of its inconsistency can simply be evaluated in terms of the rate of inconsistent pairs to all pairs.

In that respect, for assessing the level of inconsistency of a data set with CIL, it is necessary to find the number of cases associated with (R1) and (R2). To that end, the similarity or dissimilarity of each project pair $\mathbf{p}_{ij}$ concerning (1) the target variable f_{m^*} and (2) estimator variables f_m need to be judged.

[7]In other words, Equation 3.6 applies a MinMax normalization on the rank given by Equation 3.5.

Judging similarity of the target variable

The similarity of a project pair $\mathbf{p}_{ij}$ in terms of the target variable f_{m^*} is assessed based on their *relative distance*. Specifically, we denote the relative distance of a project pair $\mathbf{p}_{ij}$ in terms of the target variable f_{m^*} with $d_R(\mathbf{p}_{ij}, f_{m^*})$,

$$d_R(\mathbf{p}_{ij}, f_{m^*}) = \frac{|\mathbf{p}_i[f_{m^*}] - \mathbf{p}_j[f_{m^*}]|}{\frac{\mathbf{p}_i[f_{m^*}]+\mathbf{p}_j[f_{m^*}]}{2}}. \tag{3.7}$$

If $d_R(\mathbf{p}_{ij}, f_{m^*})$ is smaller than 1, i.e.

$$d_R(\mathbf{p}_{ij},\ f_{m^*}) < 1, \tag{3.8}$$

then the project pair $\mathbf{p}_{ij}$ is considered to have *similar* target variables. Otherwise (i.e. $d_R(\mathbf{p}_{ij},\ f_{m^*}) \geq 1$), it is regarded to have *dissimilar* target variables.

Let $\overset{\approx}{P}_{D,m^*}$ denote the set of project pairs in D with similar target variable f_{m^*}:

$$\overset{\approx}{P}_{D,m^*} = \{\mathbf{p}_{ij} \mid d_R(\mathbf{p}_{ij},\ f_{m^*}) < 1,\ \mathbf{p}_{ij} \in \mathbf{P}_D\}. \tag{3.9}$$

Moreover, let $\overset{\not\approx}{P}_{D,m^*}$ denote the set of project pairs with dissimilar target variables f_{m^*}:

$$\overset{\not\approx}{P}_{D,m^*} = \{\mathbf{p}_{ij} \mid d_R(\mathbf{p}_{ij}, m^*) \geq 1,\ \mathbf{p}_{ij} \in \mathbf{P}_D\}. \tag{3.10}$$

Judging similarity of estimator variables

The similarity of a project pair $\mathbf{p}_{ij}$ in terms of the estimator variables f_m is assessed based on their normalized rank of relative similarity d_{NR}.

Let $\overset{\approx}{P}_{D,m}$ denote the set of project pairs with similar estimator variables f_m.

$$\overset{\approx}{P}_{D,m} = \{\mathbf{p}_{ij} \mid d_{NR}(\mathbf{p}_{ij}, f_m) < \alpha,\ \mathbf{p}_{ij} \in \mathbf{P}_D\}, \tag{3.11}$$

where α is a threshold in the interval between 0 and 1. In addition, suppose that $\overset{\not\approx}{P}_{D,m}$ is the set of project pairs with dissimilar estimator variables f_m.

$$\overset{\not\approx}{P}_{D,m} = \{\mathbf{p}_{ij} \mid d_{NR}(\mathbf{p}_{ij}, f_m) \geq 1-\alpha,\ \mathbf{p}_{ij} \in \mathbf{P}_D\}. \tag{3.12}$$

Note that when $\alpha \leq d_{NR}(\mathbf{p}_{ij}, f_m) < 1-\alpha$ we do not regard the project pair $\mathbf{p}_{ij}$ to be neither similar nor dissimilar in terms of the estimator variables.

Explicit formulation of CIL

Let $\mathbf{P}_{D,R1}$ denote the set of inconsistent project pairs of D, which satisfy (R1). Then, $\mathbf{P}_{D,R1}$ can be written as

$$\mathbf{P}_{D,R1} = \left\{ \mathbf{p}_{ij} \middle| \ \mathbf{p}_{ij} \in \overset{\not\approx}{P}_{D,m^*}, \ \mathbf{p}_{ij} \in \overset{\approx}{P}_{D,m} \right\}. \tag{3.13}$$

Let $\mathbf{P}_{D,R2}$ denote the set of inconsistent project pairs of D, which satisfy (R2). $\mathbf{P}_{D,R2}$ is simply

$$\mathbf{P}_{D,R2} = \left\{ \mathbf{p}_{ij} \middle| \ \mathbf{p}_{ij} \in \overset{\approx}{P}_{D,m^*}, \ \mathbf{p}_{ij} \in \overset{\not\approx}{P}_{D,m} \right\}. \tag{3.14}$$

Putting together the information on similarity/dissimilarity of the target variable and estimator variables, CIL concerning the data set D can be expressed as

$$\mathbf{CIL}(D) = \frac{\#\left(\mathbf{P}_{D,R1} \bigcup \mathbf{P}_{D,R2}\right)}{\#(\mathbf{P}_D)}. \tag{3.15}$$

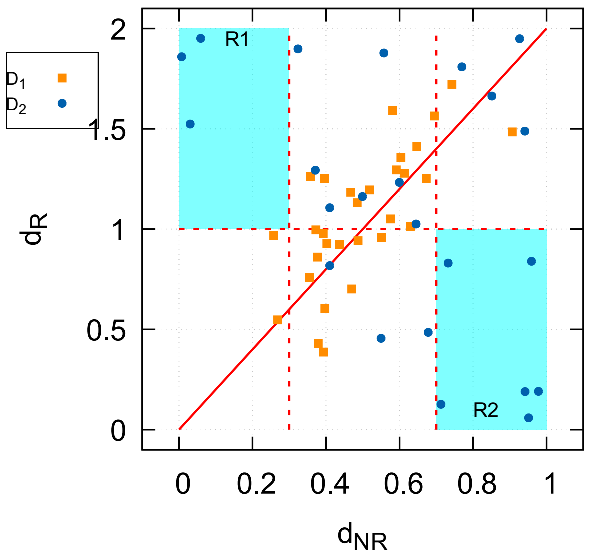


Figure 3.1: Distribution of relative distance d_R of the target variable f_{m^*} and normed rank distance d_{NR} of estimator variables f_m concerning two hypothetical data sets D_1 and D_2. The solid curve shows the linear fit for both data sets. The vertical dashed lines illustrate $d_{NR} = \alpha$ and $d_{NR} = 1 - \alpha$ for $\alpha = 0.3$ and the horizontal dashed line marks $d_R = 1$.

For two hypothetical data sets D_1 and D_2, Figure 3.1 illustrates the distribution of relative distance d_R of target variables f_{m^*} and normed rank distance d_{NR} of estimator variables f_m. Here, the shaded region in the upper left corner corresponds to (R1) and the region in the lower right corner corresponds to (R2) mentioned in Section 3.3.2.

One may see in Figure 3.1 that there are no project pairs in D_1, which meet the conditions stated in (R1) and (R2). Thus, D_1 is free of inconsistencies and has high data quality. On the other hand, there are a non-zero number of cases in D_2 corresponding to (R1) and (R2). Such inconsistencies indicate that D_2 has a lower data quality than D_1.

3.4 Follow-Up Evaluation of CIL

In the follow-up evaluation of CIL, we keep certain properties of evaluation same as Phannachitta *et al.* [78] and change certain others to get a better insight into the performance of CIL. In particular, the estimation target variable, estimator variables and performance evaluation metrics are kept the same to have a fair comparison with the original study [78][8]. On the other hand, the number and variety of data sets are increased to provide a more comprehensive evaluation. In addition, the experiment procedure is modified by diversifying experimental runs with cross-validation and testing with various values of the threshold α. In this section, we first elaborate on each of these experimental factors and then present experimental results.

3.4.1 Data sets and pre-processing

In [78], four data sets are employed in assessing the efficiency of CIL, where one particular data set, i.e. Kemerer [40], is a very small one, containing only 15 projects.

Here, in this follow-up evaluation of CIL, we conduct more extensive experiments using six reference data sets shown in Table 3.2. Note that these data sets are published as part of the SeaCraft repository [70] and are relatively large, containing at least 48 projects. And China data set contains projects from many Chinese companies, Coc81dem and Nasa93 data sets contain NASA projects, Desharnais data set contains projects from a Canadian company, Maxwell data set contains projects from banks in Finland, Miyazaki data set contains projects developed in COBOL language.

[8]Note that in the experiments, we use $C = 5$ as recommended in the original IVDM study [93].

Table 3.2 Reference data sets employed in this study.

Data set	# of projects N	# of project features
China [70]	499	12
Coc81dem [6]	63	18
Desharnais [20]	77	9
Maxwell [64]	62	26
Miyazaki94 [72]	48	8
Nasa93 [70]	93	26

As for pre-processing, no particular operation is carried out following the same strategy of [78]. Note that this helps us make a direct and fair comparison with the results reported in [78]. In addition, it helps us to confine our analysis to the evaluation of CIL and especially to the cases where it fails more frequently, rather than digressing the discussion towards what pre-processing operations are necessary or how they should be tuned etc.

Nevertheless, we of course recognize that there may be certain data treatment techniques, which may reflect as an improvement on the efficacy of CIL. In order to address such aspects, we choose in Section 3.6 certain pre-processing operations and apply them on both CIL and the proposed metric SCIL. In this way, we point out how much improvement can be obtained in CIL due to pre-processing, and how much the proposed metric can improve further over that.

3.4.2 Target variable and estimation model

Similar to [78], in the follow-up evaluation, we consider "effort" to be target variable and use all other variables in the data sets to estimate its value.

As for the estimation method, we used Classification and Regression Trees (CART) [53] with tree pruning based on the error rates in cross-validation, since it is discussed by Phannachitta *et al.* to be the most efficient estimator[78][9].

[9]Note that in Section 3.6.2 we reassess this claim.

3.4.3 Experiment procedure

In the experiments, we conducted 3 repetitions of 3-fold cross-validation for each data set, as illustrated in Figure 3.2. Specifically, we randomly split a source data set into 3 subsets. Then, we conduct three rounds of model construction and model evaluation, each of which employs two subsets in model construction. In each case, the remaining subset is used for evaluation. Eventually, we used 54 data samples derived from the six data sets shown in Table 3.2. The CIL values are calculated based on the fit subset in each case and their correlation with estimation error is examined.

We expect this series of experiments to mitigate the effect introduced by the random splitting (of the source data into test and fit data) on estimation results and provide a better understanding of stability of performance.

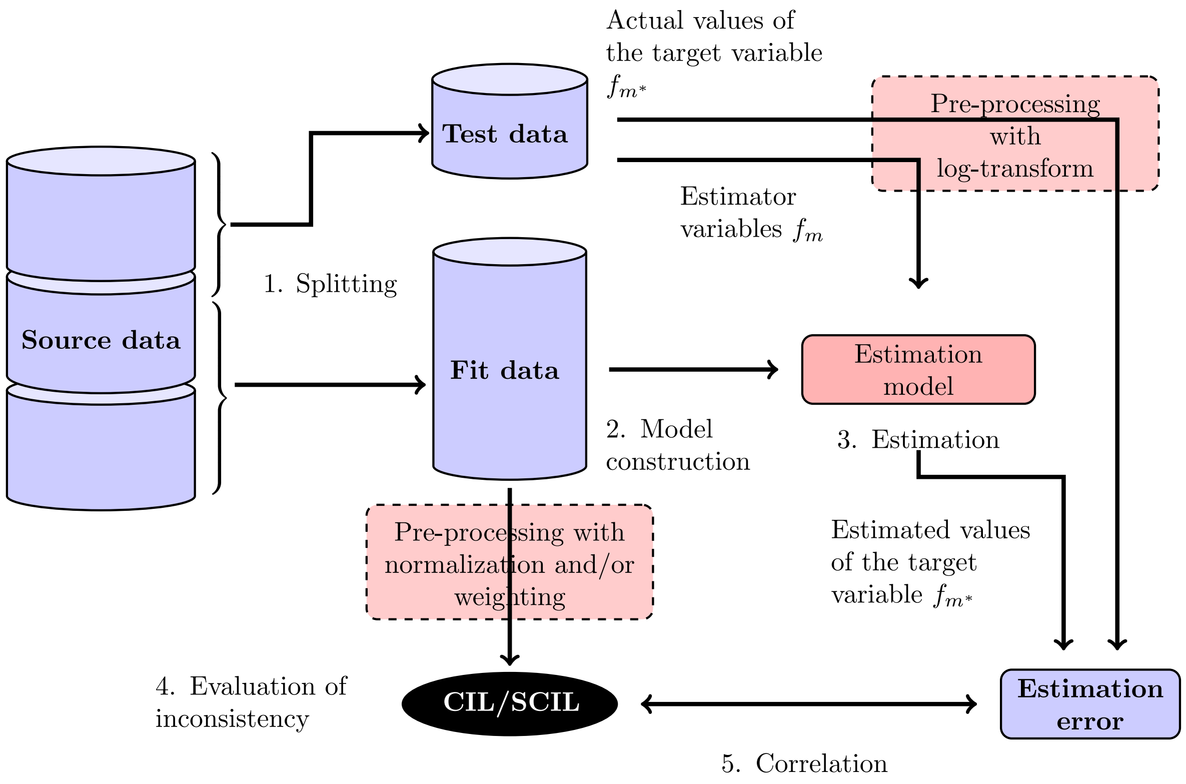

Actual values of the target variable f_{m^*}
Pre-processing with log-transform
Test data
Estimator variables f_m
Source data
1. Splitting
Fit data
Estimation model
2. Model construction
3. Estimation
Estimated values of the target variable f_{m^*}
Pre-processing with normalization and/or weighting
4. Evaluation of inconsistency
CIL/SCIL
Estimation error
5. Correlation

Figure 3.2: Experiment procedure. The blue blocks are common in all experiments, whereas the solid pink block (i.e. estimation model) varies between experiments. The dashed pink blocks are not used in the follow-up evaluation of CIL as in [78]. But they are used in used in comparing CIL and SCIL in Section 3.6.6.

3.4.4 Estimation error

For measuring estimation error, similar to Phannachitta *et al.* we use Mean Magnitude of Relative Error (MMRE) [29], which is commonly used in effort estimation studies. In particular, we denote the Magnitude of Relative Error (MRE) of project $\mathbf{p}_i$ concerning an estimation target variable of f_{m^*} with $\mathbf{MRE}(\mathbf{p}_i[f_{m^*}])$ and compute it as,

$$\mathbf{MRE}(\mathbf{p}_i[f_{m^*}]) = \frac{|\mathbf{p}_i[f_{m^*}] - \widehat{\mathbf{p}}_i[f_{m^*}]|}{\mathbf{p}_i[f_{m^*}]}, \tag{3.16}$$

where $\widehat{\mathbf{p}}_i[f_{m^*}]$ denotes the estimated value of f_{m^*} for project $\mathbf{p}_i$. Then, MMRE concerning a data set D is computed as the mean value of MREs concerning all projects in D,

$$\mathbf{MMRE}(D) = \frac{1}{\#(D)} \left(\sum_{\mathbf{p}_i \in D} \mathbf{MRE}\left(\mathbf{p}_i[f_{m^*}]\right) \right). \tag{3.17}$$

3.4.5 Assessment of efficacy of CIL

The efficacy of CIL is assessed based on its correlation with MMRE. Namely, if a data set is of high quality, then the rate of data points satisfying (R1) and (R2) should be small, yielding a low CIL. Similarly, for a data set of high quality, the estimation error (i.e. MMRE) should be small. On the contrary, a low quality data set is expected to have more data points satisfying (R1) and (R2), thus to have a higher CIL, and also to suffer from high estimation error (i.e. high MMRE).

Therefore, CIL and MMRE are expected to be positively correlated. Specifically, the correlation between CIL and MMRE is represented with R and computed as

$$R = \frac{\mathbf{Cov}(\mathbf{CIL}, \mathbf{MMRE})}{\sigma_{\mathbf{CIL}} \sigma_{\mathbf{MMRE}}}. \tag{3.18}$$

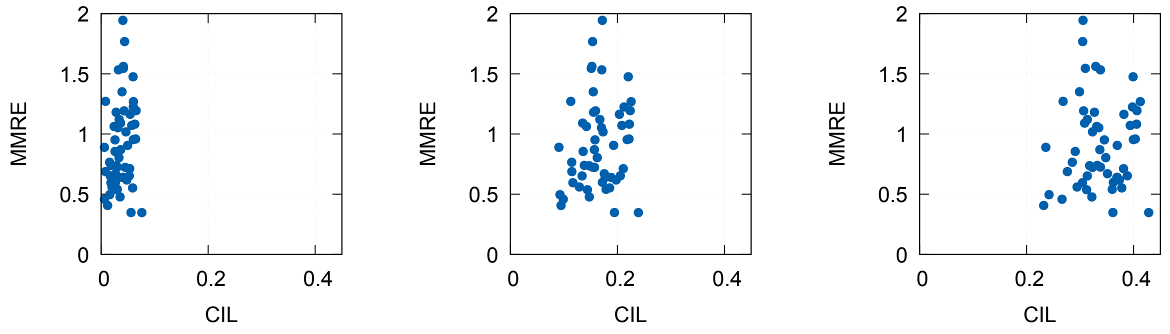
3.4.6 Results of the follow-up evaluation

Figure 3.3 and Table 3.3 show the results of our follow-up evaluation. By examining the distribution of CIL and MMRE values in Figure 3.3, one may judge in a qualitative way that there is no strong correlation between them for any of the α values.

In addition, Table 3.3 proves in a quantitative way that there is very little correlation between CIL and MMRE ($R < 0.4$) in all cases. In that respect, the current form of CIL is shown not to be effective in assessing data quality of software project data sets.

Table 3.3: The correlation coefficients R concerning **MMRE** and **CIL** values in the follow-up evaluation (without pre-processing).

Data set	R		
	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$
China [70]	-0.170	0.106	0.058
Coc81dem [6]	-0.159	-0.120	-0.439
Desharnais [20]	-0.768	-0.578	-0.640
Maxwell [64]	-0.737	-0.431	-0.265
Miyazaki94 [72]	0.194	0.387	0.261
Nasa93 [70]	-0.308	-0.011	-0.110



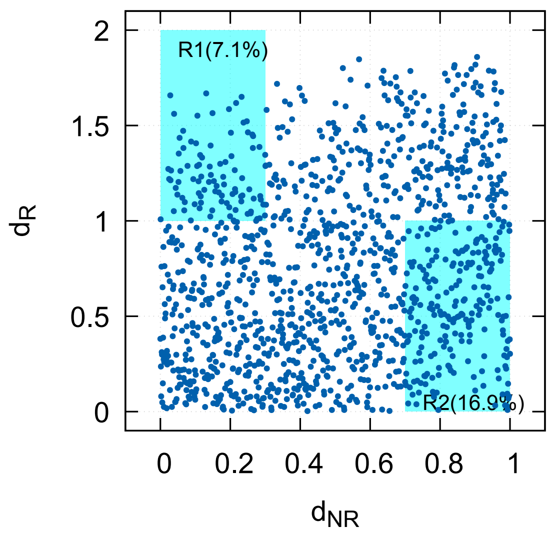
(a) (b) (c)

Figure 3.3: The relationship between CIL and MMRE in effort estimation with thresholds of (a) $\alpha = 0.1$, (b) $\alpha = 0.3$ and (c) $\alpha = 0.5$.

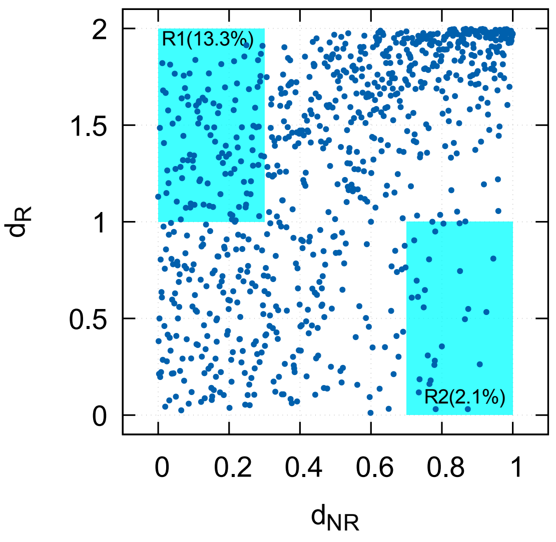
3.5 Definition of SCIL

In order to improve the efficacy of CIL, we first contemplate on the reasons for its poor performance in the follow-up analysis. To that end, we focus on two particular data sets, namely Desharnais [20] and Coc81dem [6], and take a closer look at their properties.

The distribution of relative distance d_R of the target variable and normed rank distance d_{NR} of estimator variables for Desharnais data set is presented in Figure 3.4-(a). For the threshold value $\alpha = 0.3$, the relating CIL value is found to be 0.239, whereas MMRE is



(a) Desharnais data set
(CIL = 0.239, MMRE = 0.347).



(b) Coc81dem data set
(CIL = 0.154, MMRE = 1.768).

Figure 3.4: The distribution of relative distance d_R of the target variable and normed rank distance d_{NR} of estimator variables for (a) Desharnais and (b) Coc81dem data sets. These values are obtained with 3-fold cross-validation and $\alpha = 0.3$.

found to be 0.347.

The distribution of relative distance d_R of the target variable and normed rank distance d_{NR} of estimator variables for Coc81dem data set is presented in Figure 3.4-(b). Here, the CIL and MMRE values are found to be 0.154 and 1.768, respectively, for the same threshold value $\alpha = 0.3$.

Based on CIL values, Coc81dem seems to have better data quality than Deshairnais. However, based on the MMRE values, Deshairnais seems to have a superior data quality over Coc81dem, a contradicting conclusion to CIL.

The reason for this contraction is considered to be the assumption of CIL that the contributions of the data points lying in the two regions of (R1) and (R2) to deterioration of data quality are virtually the same. We claim that the data in (R1) and (R2) contribute to data inconsistency in different ways and elaborate on our reasoning based on Figure 3.4.

The (R1) region, which corresponds to the projects with similar target variables and dissimilar estimator variables, accommodates 7.1% of the data points in Figure 3.4-(a) , whereas it accommodates 13.3% of the data points in Figure 3.4-(b). On the other hand, the (R2) region, which corresponds to the projects with similar target values but dissimilar estimator variables, accommodates 16.9% of the data in Figure 3.4-(a), whereas it accommodates 2.1% of the data in Figure 3.4-(b). In other words, the percentage of data points in (R2) of Figure 3.4-(a) (16.9%) is much larger than that of Figure 3.4-(b) (2.1%).

As a matter of fact, concerning effort, it is not uncommon that dissimilar estimator variables have very similar effort values. This indicates that it is not appropriate to focus on (R2) region in evaluating data inconsistency and that it is better to pay regard rather to (R1) region.

Based on such contemplation, we propose improving CIL by focusing on inconsistencies due to data points in the (R1) region. We call the improved metric Similar Case Inconsistency Level (SCIL) and define it explicitly as follows:

$$\mathbf{SCIL}(D) = \frac{\#\left(\mathbf{P}_{D,R1}\right)}{\#(\mathbf{P}_D)}. \tag{3.19}$$

3.6 Performance comparison of CIL and SCIL

In this section, we carry out a new performance of assessment for CIL taking in consideration the impact of several data pre-processing operations. In addition, we test SCIL with exactly the same conditions concerning this secondary re-assessment of CIL and ensure an objective (unbiased) comparison.

3.6.1 Data sets and pre-processing

In evaluating the efficacy of the proposed metric SCIL and comparing it to that of CIL, we used the data sets reported in Table 3.2 and the experiment procedure with 3-fold cross-validation defined in Section 3.4.3 (see also Figure 3.2).

However, unlike [78] and Section 3.4,we considered three kinds of pre-processing operations and applied them on the data set before computing CIL as well as SCIL. Specifically, the pre-processing operations are normalization, weighting and log-transformation, which are detailed as follows, respectively.

Normalization of features

Although normalization is a crucial part of data pre-processing in software analytics, Phannachitta *et al.* do not consider any normalization in [78]. However, the data sets under investigation contain variables from significantly different value ranges. In that respect, normalization is necessary to make sure that the project variables have a balanced influence on the calculation of distance metrics. In this chapter, we consider two alternatives for normalization, i.e. MinMax normalization and Z-score normalization (also known as standardization, see Appendix A.2) for determining the more effective normalization scheme for the target variable in focus.

Weighting

Weighting is another common data pre-processing tool in the analysis of software industry data and is omitted in [78]. Yet, it is plausible that different estimator variables are likely to have different influence on the target variable, which can be dealt with by asserting different weights on them through the correlation coefficient. In our study, we opt for weighting the estimator variables as follows:

$$\mathbf{weighted}(\mathbf{p}_i[f_m]) = \mathbf{p}_i[f_m] \times \underset{D}{\mathbf{Corr}}(f_{m^*}, f_m), \tag{3.20}$$

where $\underset{D}{\mathbf{Corr}}(f_{m^*}, f_m)$ is Pearson's correlation coefficient between the estimation target f_{m^*} and estimator variable f_m in data set D.

Note that in analyzing the effect of weighting in Section 3.6.6, we report the results of the experiments with and without weighting and point out the benefits and drawbacks.

Log transform

Kitchenham and Mendes empirically showed that logarithmic transformation helps in improving effort estimation accuracy[48]. Thus, we decided to perform experiments involving also a logarithmic transformation preceding estimation of target variables. Note that when we employ logarithmic transformation, it is applied on both the target variable and the estimator variables prior to model construction[10]. In Section 3.6.2 we compare the performance of several estimation schemes with and without log transformation, and select the best one to be deployed in the experiments.

[10]For variables containing 0, the offset value 1.0 is added before the transformation.

3.6.2 Target variables and estimation models

As mentioned in Section 3.2, it is common in ESE to use “effort” and/or “productivity” as target variables in analyzing software project data sets. In that respect, in investigating the efficacy of SCIL and comparing it to that of CIL, we diversify target variables by considering both “effort” and “productivity”.

While effort refers to the amount of the professional activity in man-hours to complete a project (e.g. by developers, testers etc.), productivity is generally defined as the size of development per unit effort. Specifically, we employ the following definition for productivity,

$$\text{Productivity} = \frac{FP}{\text{Effort}}, \tag{3.21}$$

where FP stands for the “function point”.

Note that not all data sets in Table 3.2 involve FP as a project feature. However, these sets involve “lines of code” (LOC), which can be used as a replacement in Equation 3.21 (for the purpose of this study). In that respect, if the data set under investigation involves FP as a project feature, we compute productivity as in Equation 3.21, and otherwise we replace FP with LOC.

In effort estimation studies, linear regression technique is commonly used [45, 48]. Besides, CART with tree pruning based on the error rates in cross-validation [53] (hereafter denoted as CART + Tree pruning) is shown to be the best method in the original CIL study [78]. Furthermore, Random Forest technique is shown to be a promising method in recent effort estimation studies [4, 8, 90]. Therefore, we employ all these models and select the model with the smallest estimation error to evaluate and compare CIL and SCIL.

Table 3.4 shows the result of effort estimation of the above models with/without logarithmic transformation[11]. According to Table 3.4, the minimum estimation error is obtained by the Random Forest technique with logarithmic transformation. Note that as mentioned in Section 3.4.2, Phannachitta *et al.* claim the best performing estimator to be CART + Tree pruning in [78]. However, based on the results presented in Table 3.4, we observe that Random Forest with logarithmic transformation performs even better. Therefore, in Section 3.6.6, we carry out performance evaluation and comparison of CIL and SCIL based on Random Forest technique with logarithmic transformation.

Finally, we note that the integration of pre-processing operations and the improvement

[11]Note that + and - denote an experiment with and without log-transform, respectively.

Table 3.4: The ranking of effort estimation models with respect to ascending values of MMRE.

Rank	Log transform	Estimation model	Average of MMRE
1	+	Random Forest	**0.660**
2	+	Linear regression	0.687
3	+	CART + Tree pruning	0.912
4	-	Random Forest	1.439
5	-	CART + Tree pruning	2.228
6	-	Linear regression	4.638

of the estimator model are expected to enhance the performance of CIL reported in Section 3.6.6. In order to provide an insight how much these two modifications contribute to it, in Appendix A.3, we first integrate a pre-processing module to the original framework of CIL and then change the former estimator (CART+Tree pruning) with a more competent one (Random Forest). By this means, we assess the improvement that can be expected on CIL by applying such extras/fine-tuning.

3.6.3 Experiment procedure

Similar to Section 3.4, we repeat 3-fold cross-validation 3 times yielding 54 different experimental runs (see also Section 3.4.3). In addition, we employ different combinations of distance functions, normalization techniques and weighting or not to find which combination works best, and ensure a comprehensive assessment.

As explained in Section 3.3.2, Phannachitta *et al.* employ IVDM in computing $d_{NR}(\mathbf{p}_{ij}, f_{m^*})$. However, it is not clear whether IVDM is the best distance function to evaluate relative difference of projects or whether alternative metrics can perform better. Therefore, we compute the proposed SCIL metric as well as the previously proposed CIL metric using three different distance functions: Euclidean Distance d_E, cosine distance d_C (see Appendix A.1), and IVDM. We contrast the performance values for figuring out which distance function is most appropriate.

In addition, at each run three different threshold values $\alpha \in \{0.1, 0.3, 0, 5\}$ are used in computing d_{NR}. Namely, the similarity or dissimilarity of estimator variables are judged at varying degrees. Namely, higher values of α imply a broad range for inconsistency (for that matter also consistency), whereas lower values of α leave more space for ambiguity (i.e. regarding the estimator variables as neither similar nor dissimilar).

3.6.4 Estimation error

In assessing the performance of CIL and SCIL, we use mean MMRE[12] as the estimation error similar to the follow-up evaluation of CIL reported in Section 3.4.

3.6.5 Assessment of efficacy of SCIL

Based on the apprehension that a more consistent data set is likely to contribute to a more accurate estimation model, we evaluate CIL and SCIL metrics by observing the correlation between them and estimation errors concerning the models built from real project data sets. We expect that the data sets with higher data quality (i.e. characterized by lower values of CIL or SCIL) will on average have a lower estimation error than that of the data sets with lower data quality[13].

3.6.6 Results on comparison of CIL and SCIL

In this section, we assess the performance of CIL and SCIL metrics from the point of view of the "fitness for purpose" [58], where the "purpose" is determined as "effort estimation" and "productivity estimation" within the scope of this study. To that end, we analyze the correlation R between the estimation error (of effort and productivity) quantified in terms of MMRE and the values of two data inconsistency metrics CIL and SCIL. The following subsections presents the results relating to effort estimation and productivity estimation, respectively.

[12]Note that here "mean" refers to the average over all data sets illustrated in Table 3.2.

[13]In other words, CIL/SCIL values are expected to be positively correlated with estimation error, where a higher correlation indicates a better assessment of data quality.

Effort estimation

Concerning effort estimation, Tables 3.5 and 3.6 present the correlation of MMRE with CIL and SCIL, respectively[14]. Note that in these tables as pre-processing scheme we consider different combinations of distance metrics, normalization and weighting schemes.

From Tables 3.5 and 3.6, we can see that the correlation of SCIL and MMRE is considerably better than that of CIL and MMRE for any data pre-processing scheme. In other words, between the corresponding values of CIL and SCIL in the two tables, the one of SCIL is always larger.

In addition to serving for performance comparison, Tables 3.5 and 3.6 help us also identify an efficient pre-processing scheme for CIL and SCIL[15]. In that respect, we start by taking a closer look at Table 3.6, since the main focus of our study is SCIL. The values in bold are highest in their corresponding rows. In addition, the highlighted row corresponding to

Table 3.5 Correlation between **MMRE** and **CIL** concerning target variable of *effort*.

Distance	Norm.	Weigh.	R		
			$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$
d_E	Z-score	+	**0.459**	0.439	0.320
d_E	Z-score	-	0.090	**0.285**	0.213
d_E	MinMax	+	0.450	0.479	**0.494**
d_E	MinMax	-	0.142	0.204	**0.265**
d_C	Z-score	+	-0.127	**0.049**	0.048
d_C	Z-score	-	-0.117	-0.010	**0.046**
d_C	MinMax	+	**-0.315**	-0.368	-0.338
d_C	MinMax	-	**-0.157**	-0.192	-0.162
IVDM	*	*	**0.381**	0.372	0.290

[14]Note that correlation is computed in the same way as in Equation 3.18, but by replacing CIL with SCIL. Note also that + and - denote an experiment with and without weighting, respectively, whereas * denotes that normalization or weighting does not apply to IVDM.

[15]Since virtually there are an infinite number of pre-processing possibilities, we can not claim that we identified the *optimal* scheme. Nevertheless, we can say that we have found one with a fairly good performance.

Table 3.6 Correlation between **MMRE** and **SCIL** concerning target variable of *effort*.

Distance	Norm.	Weigh.	R		
			$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$
d_E	Z-score	+	0.604	**0.615**	0.570
d_E	Z-score	-	0.531	**0.534**	0.485
d_E	MinMax	+	**0.578**	0.575	**0.578**
d_E	MinMax	-	0.527	**0.530**	0.524
d_C	Z-score	+	0.391	**0.472**	0.456
d_C	Z-score	-	0.486	**0.522**	0.515
d_C	MinMax	+	0.079	0.218	**0.340**
d_C	MinMax	-	0.361	0.392	**0.424**
IVDM	*	*	**0.629**	0.624	0.583

IVDM attains the highest correlation overall ($R = 0.629$) with $\alpha = 0.1$. Actually, IVDM attains the highest value for any threshold value of α (i.e. in every column of Table 3.6). In addition, Euclidean distance d_E coupled with Z-score normalization and weighting is observed to attain comparable results to those of IVDM concerning all α values.

Regarding the variation on MMRE due to variations on α, we can say that 0.1 and 0.3 are superior to 0.5, since both IVDM and Euclidean distance d_E coupled with Z-score normalization and weighting attain $R > 0.6$, indicating a relatively high correlation between SCIL and MMRE (see Table 3.6). Overall, to compute SCIL metric for effort estimation purposes, IVDM or the Euclidean distance d_E coupled with Z-score normalization and weighting and $\alpha = 0.1$ or 0.3 can be recommended.

On the other hand, cosine distance d_C coupled with MinMax normalization and weighting shows the lowest correlation for any threshold value α. Note also that cosine distance d_C shows in general lower correlation (i.e. regardless of the pre-processing techniques) and therefore, it is not recommended to be used for the purpose of effort estimation.

Next, we also take a closer look at the results concerning CIL in Table 3.5. We can see that the values at the highlighted row are quite high in their corresponding columns (actually, the highest for $\alpha = 0.3$ and $\alpha = 0.5$ and very close to the highest for $\alpha = 0.1$). In that respect, the best performing combination of distance, normalization and weighting scheme is found to be Euclidean distance d_E coupled with MinMax normalization and weighting. Also, replacing MinMax normalization with Z-score normalization in this combination yields somewhat comparable results.

Productivity estimation

Concerning productivity estimation, Tables 3.7 and 3.8 present the correlation of MMRE with CIL and SCIL, respectively. We can see that the relation between Tables 3.7 and 3.8 is similar to the relation between Tables 3.5 and 3.6. Namely, similar to effort, also for productivity estimation, the correlation between SCIL and MMRE is considerably better than that of CIL and MMRE for any data pre-processing scheme. Specifically, CIL attains the highest value of $R = 0.518$, whereas SCIL gets a maximum of $R = 0.814$. Moreover, the highest performance concerning different combinations of pre-processing schemes (i.e. bold values in Table 3.7) varies considerably for CIL (i.e. between -0.337 and 0.518), whereas for SCIL they are somewhat more stable (i.e. 0.564 and 0.819). In addition, between the corresponding values of CIL and SCIL in Tables 3.7 and 3.8, the one of SCIL is always larger.

Next, we take a closer look at Table 3.8 for identifying the best performing combination

Table 3.7: Correlation between **MMRE** and **CIL** concerning target variable of *productivity*.

Distance	Norm.	Weight.	R		
			$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$
d_E	Z-score	+	**0.509**	0.388	0.356
d_E	Z-score	-	0.354	0.331	**0.437**
d_E	MinMax	+	**0.518**	0.384	0.338
d_E	MinMax	-	0.333	0.376	**0.398**
d_C	Z-score	+	-0.394	-0.374	**-0.337**
d_C	Z-score	-	-0.116	-0.169	**-0.095**
d_C	MinMax	+	**-0.139**	-0.140	-0.156
d_C	MinMax	-	**0.015**	-0.127	-0.126
IVDM	*	*	**0.094**	-0.257	-0.313

Table 3.8: Correlation between **MMRE** and **SCIL** concerning target variable of *productivity.*

Distance	Norm.	Weight.	R		
			$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$
d_E	Z-score	+	0.773	0.805	**0.814**
d_E	Z-score	-	0.721	0.790	**0.804**
d_E	MinMax	+	0.781	0.802	**0.814**
d_E	MinMax	-	0.731	0.808	**0.819**
d_C	Z-score	+	0.094	0.294	**0.564**
d_C	Z-score	-	0.463	0.592	**0.670**
d_C	MinMax	+	0.174	0.358	**0.558**
d_C	MinMax	-	0.391	0.518	**0.635**
IVDM	*	*	0.673	0.688	**0.698**

of pre-processing operations concerning SCIL. Unlike the results for effort estimation, the Euclidean distance d_E attains the highest correlation values regarding all three thresholds α, regardless of the normalization technique or weighting. The IVDM performs second best, and cosine distance d_C performs worst regardless of pre-processing.

Regarding the variation on MMRE due to variations on α, 0.3 or 0.5 are recommended in computing SCIL, since in all cases with Euclidean distance d_E, $R > 0.8$ is attained, which indicates to a high correlation between SCIL and MMRE. Taking in consideration also the results of effort estimation reported in Table 3.6, Euclidean distance function coupled with Z-score normalization and weighting with $\alpha = 0.3$ is highly recommended to compute SCIL metric for both effort and productivity estimation purposes.

Subsequently, we take a closer look at the results concerning CIL. In Table 3.7, we can see that Euclidean distance d_E coupled with MinMax normalization and weighting attains the highest value of correlation ($R = 0.518$). In that respect, the best performing pre-processing combination is exactly the same as the one for estimating effort (see Table 3.5). Moreover, similar to effort estimation, replacing MinMax normalization with Z-score normalization again yields somewhat comparable results.

Finally, we would like to draw the attention of the reader to the similarity of the distribution of the highest values in each row of Tables 3.5 and 3.7 and also Tables 3.6 and 3.8. One may see that the pattern is quite similar between Tables 3.5 and 3.7, whereas very different between Tables 3.6 and 3.8. We believe that this indicates the prominent effect of pre-processing on the performance of CIL. In other words, CIL is quite sensitive to the changes in experimental conditions and is likely to miss the actual role of input and/or target in estimation accuracy.

Based on the above discussion, we conclude that the proposed metric SCIL effectively quantifies the level of inconsistency of a data set, is considerably superior to CIL and appraises data inconsistency in a more resilient manner.

3.7 Threats to validity

We provide a discussion on the validity of the proposed method in terms of three commonly adopted experimental validation approaches, i.e. internal validity, external validity and construct validity.

Internal validity refers to the extent by which the observed effect is a consequence of the presumed cause. In our case, internal validity questions whether or not different conclusions can be drawn with regard to the different settings in the experiment. To ensure internal validity, we conducted 3 repetitions in the validation process to produce stable results. However, there is one possible issue of internal validity in this study. The issue is the single sampling method (3-fold cross-validation) we used. Our important future work is to employ other methods such as leave-one-out cross-validation to increase the validity of the result.

External validity refers to the generalization of the results. In this study, we address external validity by using 6 reference data sets with diverse characteristics. Namely, they vary in size (i.e. number of projects), and project variables, as well as origin (i.e. recording organization) and recording period. Our future work is to employ more data sets to increase the generalization of the results.

Construct validity refers to the relevance and capability of the observations and measurements in evaluating the posed hypothesis. In this study, we use single error measure MMRE to evaluate the target variable value estimation performance. It is our future work to employ other error measures to increase the validity of our work.

3.8 Conclusions and future prospects

This study proposes an improved data quality metric SCIL that can quantify the level of inconsistency of a data set. Comparing the conventional CIL metric and the SCIL metric, we believe that the proposed SCIL metric is more suitable for the purpose of effort and productivity estimation as we have shown through experimental evaluation. Considering the experimental results, it is recommended to combine the Euclidean distance function be combined with the Z-score normalization and weighting (threshold $\alpha = 0.3$) to calculate the SCIL metric.

As future work, we will employ other methods such as leave-one-out cross-validation and other error measures to increase validity, and more data sets to increase generality.

Chapter 4

Neg/pos-normalized Accuracy Measures for Defect Prediction

4.1 Introduction and motivation

To improve the efficiency of software testing and/or maintenance, it is important to predict defect-prone modules and assign more effort to them. To this end, defect-prone module prediction (or simply, defect prediction) has been studied over decades by many researchers [5, 36, 38, 39, 42, 44, 47, 50, 54, 67].

In assessing the accuracy of prediction models, it is common to use Precision, Recall, F1-value, Balance [66], Area Under the Curve (AUC) of Receiver Operating Characteristics (ROC) [7, 34], etc. However, there is no consensus on how to interpret these accuracy values. For example, a model with a precision of 0.8 seems to achieve a pretty good performance, but can we confidently claim that this model is successful? The decision will depend on the data set. If the data set contains many defects (e.g. 80% of the modules are defect-prone), a precision of 0.8 is not sufficient for calling that prediction model successful. Namely, we can approach such an accuracy even by assigning classes randomly, whereas, this is not the case for data sets with very few defects. In that respect, the susceptibility of the accuracy measures to the proportion of defect-prone and defect-free modules in the data set is a serious problem [65].

In order to address this issue, we propose deploying the *expected values* of accuracy measures, which we show to be possible to compute solely from the number of positive and negative instances in the data set. In this way, if the (raw) value of a certain accuracy measure is larger than its expected value, the relating model is considered to be a successful one and a proper action is suggested to be taken according to its outcomes.

We deploy these expected values in defining *neg/pos-normalized values* of various accuracy measures, which enables us to rank predictions across different data sets. Accounting for neg/pos ratios of the data sets, we can assess the performance of the prediction model under investigation in a more objective way.

In order to demonstrate the capabilities of the proposed approach, we carry out a case study on 38 releases of 19 open source software (OSS) project data sets from 3 data sources. Specifically, we first carry out defect prediction on these sets and evaluate performance in terms of several conventional accuracy measures. We then compare them with their corresponding expected values. The case study confirms that the predictions, which yield higher accuracy than respective expected values are indeed successful.

This chapter is organized as follows. In Section 4.2, we elaborate on existing accuracy measures pointing out their shortcomings. We also justify the reason for establishing baseline values for accuracy measures through a preliminary analysis of 19 defect prediction data sets. In Section 4.3 we first introduce the basic idea of the proposed approach and then provide explicit definitions. Subsequently, in Section 4.4 we present a case study of defect prediction to demonstrate how well the proposed measures address the susceptibility of the accuracy measures to the proportion of defect-prone and defect-free modules in the data set. Subsequently, in Section 4.5 we provide threats to the validity of our work and summarize our main results, contributions and future work in Section 4.6.

4.2 Background

In binary classification tasks, it is common to evaluate performance based on a contingency table like the one illustrated in Table 4.1. Specifically, the two dimensions depict the actual and predicted states of the elements of the data set. Here, we use A^+ and A^- to denote actual positives and actual negatives, and P^+ and P^- to denote predicted positives and predicted negatives. As framed within the scope of this study, an *actual positive* is a defect-prone module, whereas an *actual negative* is a defect-free module.

The prediction results that correctly indicate the presence or absence of a defect are referred to as *True Positive* and *True Negative*, respectively. In addition, the prediction

Table 4.1 Contingency table for binary classification tasks.

		Predicted state	
		P^+	P^-
Actual state	A^+	TP	FN
	A^-	FP	TN

results, which wrongly indicate the presence of a defect, are called *False Positive* and the ones, which wrongly indicate the absence of a defect, are called *False Negative.* In Table 4.1 the number of such prediction results are denoted with TP, TN, FP, FN in the corresponding cells[1].

Based on such values, one can calculate various measures to evaluate classification performance. Some conventional measures involve Precision, Recall, NPV and Specificity[2]. Henceforth, we use the normal font to refer to an accuracy measure (e.g. Precision) and typewriter font when it is treated as a random variable or a value that it takes (e.g. `Precision` and `precision`, respectively).

In terms of the entries in Table 4.1, the above-mentioned conventional accuracy measures are computed explicitly as:

$$\texttt{precision} = \frac{TP}{TP + FP}; \tag{4.1}$$

$$\texttt{recall} = \frac{TP}{TP + FN}; \tag{4.2}$$

$$\texttt{npv} = \frac{TN}{FN + TN}; \tag{4.3}$$

$$\texttt{specificity} = \frac{TN}{TN + FP}. \tag{4.4}$$

4.2.1 Weakness of conventional accuracy measures

It is common to use Precision and Recall for evaluating performance in the detection of positive cases (i.e. defect-prone modules) [36, 44, 61], whereas NPV and Specificity are used

[1]Note that in Table 4.1 $TP, TN, FP, FN \in \mathbf{Z}_{\geq 0}$.

[2]Precision is also known as Positive Predictive Value, whereas Recall and Specificity are also referred to as True Positive Rate and True Negative Rate, respectively.

mostly for evaluating performance concerning the identification of negative cases (i.e. defect-free modules) [26, 43, 71, 84, 88, 95]. Nevertheless, there is no universal agreement on which measure is the most appropriate one for each detection task.

For various purposes such as quality assurance, Precision and Recall are the most intuitive measures, since they reflect performance in detecting positive cases (i.e. defect-prone modules). Nevertheless, as discussed by Zhang and Zhang [100], a coupled deliberation of Precision and Recall is indispensable, since there is an inherent trade-off between them. Namely, in order to increase `precision`, one may simply consider as positive only those cases with a very high probability of being positive. But this will raise FN and thus decrease `recall`. On the other hand, if the condition for being considered positive is loosened, `recall` can be improved, at the cost of deteriorating `precision`. However, even a coupled deliberation may not be sufficient in certain cases. For instance, Chicco and Jurman [19] argue that the F1-value (which is the harmonic mean of the `precision` and `recall`) can dangerously show overoptimistic inflated results, especially on imbalanced data sets.

Therefore, in addition to the detection of positive cases (i.e. Precision and Recall), one needs to pay regard also to the detection of negative cases [76, 19]. Indeed, a `recall` of 1.0 can be easily achieved by predicting all modules as positive, but this would result in poor predictive performance for negative cases, i.e. `specificity` becomes zero. In this respect, Chicco and Jurman [19] recommend using Matthews Correlation Coefficient (MCC), which considers both positive and negative prediction performance. MCC was Originally developed by Matthews in 1975 for comparison of chemical structures [63] and then re-proposed by Baldi and colleagues [9] in 2000 as a standard performance metric for machine learning with a natural extension to the multiclass case [22].

However, note that an important criticism to all such performance measures is their susceptibility to certain intrinsic features of the data. In defect prediction domain, Menzies et al. [65] argue that the accuracy measures in Equations 4.1∼4.4 as well as MCC are not robust [37, 55, 99], since they are very often affected by the *neg/pos* ratio of the data set, where *pos* is the number of actual positive (i.e. defect-prone) modules (i.e. A^+) and *neg* is the number of actual negative (i.e. defect-free) modules (i.e. A^-),

$$\text{Neg/pos ratio} = \frac{A^-}{A^+}. \tag{4.5}$$

For instance, a large neg/pos ratio tends to yield low `precision`. Nevertheless, Menzies et al. also argue that a low `precision` per se is not always a problem, since there are many cases where a low `precision` is acceptable. On the other hand, although composite measures such as AUC of ROC, Balance and G-mean are known to be more robust against neg/pos ratio [32, 30, 21], according to their definition, AUC of ROC and Balance focus solely on positive cases and ignore the negative ones, whereas G-mean [47] ignores the precision of both positive and negative cases.

In this chapter, exploiting the intuitive nature of four conventional accuracy measures of Precision, Recall, NPV and Specificity, we propose using their expected values, which can be regarded as a baseline, to overcome their susceptibility to neg/pos ratio, and to avoid over-optimistic or over-pessimistic interpretations. We also propose a neg/pos-normalization scheme to enable a fair and objective assessment of model performance, and define the "successful prediction" based on proposed neg/pos-normalization scheme to focus on both positive and negative cases, and both recall and precision. By comparing prediction results across different data sets with different neg/pos ratios, we show that the neg/pos-normalized values are more robust against the composition of the data set. We compare our measures with the F1-value, AUC of ROC, MCC, G-mean and Balance to show how these metrics give a similar or different evaluation for the same model, and show that there are some cases where these composite measures are not considered useful based on the definition of "successful prediction".

4.2.2 Preliminary analysis

To demonstrate the influence of neg/pos ratio on the assessment of defect prediction accuracy, we conduct a preliminary analysis.

Data sets and prediction method

We collected 19 open source software project data sets from 3 sources (see Table 4.2). Specifically, the data sets from no.1 to no.4 (MYLN, PDE, JDT and NBNS) are introduced by Kamei et al. and the details on data collection and measurement methods can be found in [50, 69, 94]. The data sets from no. 5 to no.13 (ANT, CAML, FRST, JEDT, LOG4, LUCN, POI, PROP, SYNP) are donated by Jureczko et al. [38, 39] to the SeaCraft repository [70] and their specifics are reported in [39]. The remaining data sets from no.14 to no.19 (ECOS,

Table 4.2: Summary of the defect data sets used in the experiments. the maximum and minimum rate of defect-prone modules are indicated in bold face with *.

No.	Project Name	Short Name	Version	Number of total modules T	Number of defect-prone modules A^+	Percentage of defect-prone modules	Neg/pos ratio A^-/A^+
1	Mylyn	MYLN	2	1230	848	68.9	0.450
			3	1502	606	40.4	1.479
2	Eclipse	PDE	3.1	228	49	21.5	3.653
			3.2	309	79	25.6	2.911
3	Eclipse	JDT	3.1	3192	528	16.5	5.045
			3.2	3408	373	10.9	8.137
4	Netbeans	NBNS	4	4660	898	19.3	4.189
			5	9332	1365	14.6	5.837
5	Ant	ANT	1.5	293	32	10.9	8.156
			1.6	350	92	26.3	2.804
6	Camel	CAML	1.4	856	144	16.8	4.944
			1.6	945	188	19.9	4.027
7	Forrest	FRST	0.7	29	5	17.2	4.800
			0.8	32	2	6.3	15.000
8	Jedit	JEDT	4.2	367	48	13.1	6.646
			4.3	492	11	2.2	43.727
9	Log4j	LOG4	1.1	109	37	33.9	1.946
			1.2	205	189	**92.2***	**0.085***
10	Lucene	LUCN	2.2	247	144	58.3	0.715
			2.4	340	203	59.7	0.675
11	Poi	POI	2.5	384	248	64.6	0.548
			3	441	281	63.7	0.569
12	Prop	PROP	4	8702	840	9.7	9.360
			5	8506	1298	15.3	5.553
13	Synapse	SYNP	1	157	16	10.2	8.813
			1.1	222	60	27	2.700
14	eCos	ECOS	2	621	110	17.7	4.645
			3	3459	67	**1.9***	**50.627***
15	Exim	EXIM	4.63	184	28	15.2	5.571
			4.68	61	31	50.8	0.968
16	Ganymede	GNY	1.0pre	99	29	29.3	2.414
			1	90	30	33.3	2.000
17	Helma	HLMA	1.3.1	145	38	26.2	2.816
			1.4.0	100	17	17	4.882
18	Hibernate	HBNT	3.6.1	374	87	23.3	3.299
			3.6.2	4878	1178	24.1	3.141
19	XDoclet	XDOC	1.2Beta2	130	5	3.8	25.000
			1.2Beta3	102	30	29.4	2.400

EXIM, GNY, HLMA, HBNT, XDOC) are collected from the Software Engineering Data Repository for Research and Education [46] and their details are elaborated on in [12] [3].

Note that each data set in Table 4.2 is represented with 2 releases, which enables a cross-version defect prediction. In our preliminary experiments, we performed such defect prediction, where the older release is used as training data and the newer release is used as test data. As for the specific prediction method, we deployed random forest, since it is shown by Lessmann et al. to constitute one of the best-performing defect prediction methods [54].

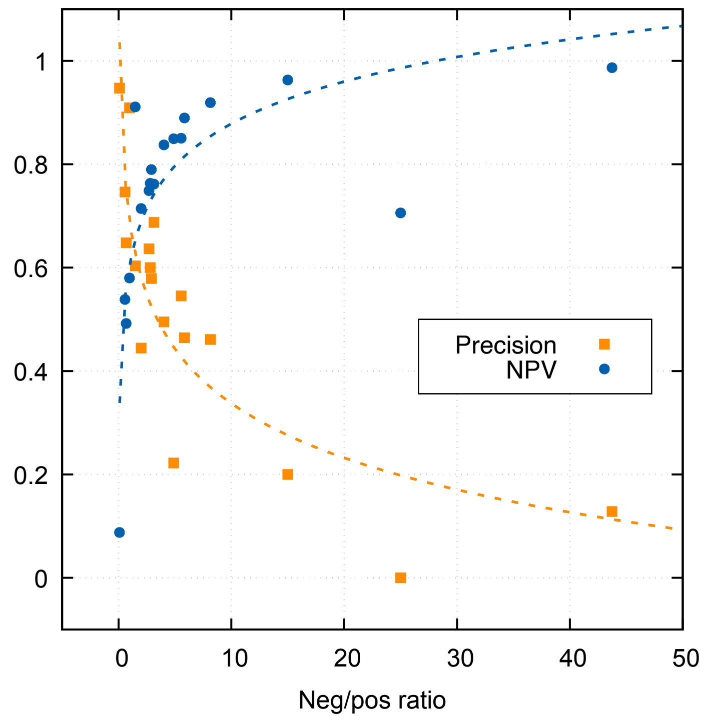


Figure 4.1: The relationship between neg/pos ratio and precision (r=-0.758) and NPV (r=0.469) in cross-version defect prediction (see Equations 4.1 and 4.3). Dashed curves represent approximations with logarithmic function and r denotes the correlation coefficient.

Results of the preliminary analysis

The results of our preliminary analysis are illustrated in Figure 4.1, where the x-axis shows the neg/pos ratio (see Equation 4.5) and the y-axis depicts the `precision` or `npv` of cross-version defect prediction[4]. From Figure 4.1, we can see clearly that `precision` decreases with increasing neg/pos ratio, whereas `npv` increases. In other words, concerning data sets with a high number of positives (i.e. defect-prone modules), `precision` may turn out to be high as a natural result of the composition of the set. Similarly, regarding data sets with a high number of negatives (i.e. defect-free modules), `npv` may unfold to be higher, merely due to the simplification of detection (of negatives).

As a result, we can say that `Precision` and `Npv` depend considerably on the composition (i.e. neg/pos ratio) of the particular data set under investigation. In that respect, judging the performance of a model based on (raw) accuracy measures without paying regard to their neg/pos ratios may lead to misleading or contradicting conclusions.

4.3 Proposed Measures

Our basic assumption is that all possible prediction outcomes with the *correct composition* have the *same likelihood* to occur. This assumption is determined paying regard to (i) fairness and (ii) desirability of prediction.

Fairness addresses the distribution of positives and negatives in the data set and the impact of that on accuracy metrics. Consider defect prediction on a data set, where actual class of all modules are positive (i.e. defect-prone). In that case, the only possible `precision` is 1.0. It is not fair to compare the performance of some model on this data set to its performance on another data set. Similarly, concerning defect prediction on a data set with all negatives (i.e. defect-free modules), the only possible `precision` is 0.0, which again precludes a fair comparison to another data set.

Desirability addresses the condition, in which the percentage of defect-prone modules in prediction should match the percentage in the test data. In practice, this is not easy to achieve, and overestimation and underestimation frequently occur due to the difference in

[3]Note that from these 19 datasets we removed those modules with zero lines of code and used the remaining modules in our analysis.

[4]In order to have a clear illustration, we chose two out of the four conventional accuracy measures in Equation 4.1∼4.4, where one focuses on positive cases and the other on negative cases.

the percentage of defect-prone modules in training and test data. To address this issue, researchers propose remedies based on numerical approaches, such as misclassification rate balancing [51], over-/under-sampling [5, 33, 68, 98], and transfer-learning [62, 96].

To satisfy fairness and desirability and to evaluate prediction models in an unprejudiced way, we believe that it is necessary to account for the proportion of defect-free and defect-prone modules in the data set. To that end, in what follows, we first discuss the expected values of accuracy measures and then introduce the proposed normalization scheme.

4.3.1 Expected values of accuracy measures

In this section, we first compute of expected values of TP, TN etc., deploy them to compute the expected values of conventional accuracy measures, explaining relating specifics on a general case, whereas a demonstration on a specific toy example is provided in Appendix B.1.

As mentioned in Section 4.3, our basic assumption is that all possible prediction outcomes with the correct composition occur with the same likelihood. Suppose that we perform defect prediction on a test data set with T modules. Let the number of actual positives and actual negatives be denoted with A^+ and A^-, respectively (see also Table 4.1). Clearly, $T = A^+ + A^-$ for actual states, and also $T = P^+ + P^-$ for predicted states.

Let an arbitrary prediction result[5] be represented with a vector π_i. Since there are T modules in the test data set, we may use a binary vector with T components (i.e. using a "1" for a predicted positive module and a "0" for a predicted negative module.).

Predicating on fairness and desirability, we consider only those vectors π_i for which $P^+ = A^+$ and $P^- = A^-$. In other words, A^+ modules are predicted as positive and A^- modules are predicted as negative, all of which are not necessarily predicted correctly. Suppose that $\Pi(A^+, A-) = \{\pi_i\}$ is the set of all such vectors (i.e. prediction results), Specifically, the elements of $\Pi(A^+, A^-)$ are permutations of A^+ "1"s and A^- "0"s.

The number of elements of $\Pi(A^+, A^-)$ is the total number of all such permutations. Namely,

$$\#\Big(\Pi(A^+, A^-)\Big) = \binom{T}{A^+}, \tag{4.6}$$

as $\#(\cdot)$ denotes the number of elements of a set and $T = A^+ + A^-$ as mentioned above.

[5] Here, the "arbitrariness" does not refer to the stochasticity of the prediction model. It simply refers to the fact that one can apply this method to any data set with a known neg/pos ratio, assuming equal probabilities for each possible prediction outcome delivered by a (black box) prediction model.

Let us focus on a particular actual positive module. There will be $\#\big(\Pi(A^+, A^-)\big)$ predictions for this module (i.e. one in each vector π_i). In addition, due to our basic assumption, A^+/T of those predictions will indeed be positive (i.e. True Positive). Thus, concerning each actual positive case, the number of positive predictions in all vectors π_i is given by

$$\frac{A^+}{T} \cdot \binom{T}{A^+}. \tag{4.7}$$

Moreover, since there are A^+ actual positive modules in the test data, the total number of positive modules, which are correctly predicted as positive (i.e. TP), is

$$TP = A^+ \cdot \frac{A^+}{T} \cdot \binom{T}{A^+}. \tag{4.8}$$

From this value, the expected value of TP can be computed as TP *per prediction.* Remember that the number of predictions (i.e. possible permutations) is as given in Equation 4.6. Thus, we get

$$\mathbf{E}(TP) = \frac{TP}{\#\big(\Pi(A^-, A^+)\big)} = \frac{\frac{A^+ \cdot A^+ \cdot \binom{T}{A^+}}{T}}{\binom{T}{A^+}} = \frac{(A^+)^2}{T}. \tag{4.9}$$

Also, the total number of FP, FN and TN can be computed in a similar way to Equation 4.8 and their respective expected values can be obtained in a similar way to Equation 4.9 [6].

Finally, such expected values can be deployed in computing the expected values of the commonly used accuracy measures. In particular, the expected values for the measures given in Equations 4.1∼4.4 can be computed as

$$\mathbf{E}(\texttt{Precision}) = \frac{\mathbf{E}(TP)}{\mathbf{E}(TP) + \mathbf{E}(FP)} = \frac{A^+}{T}, \tag{4.10}$$

$$\mathbf{E}(\texttt{Recall}) = \frac{\mathbf{E}(TP)}{\mathbf{E}(TP) + \mathbf{E}(FN)} = \frac{A^+}{T}, \tag{4.11}$$

$$\mathbf{E}(\texttt{Npv}) = \frac{\mathbf{E}(TN)}{\mathbf{E}(FN) + \mathbf{E}(TN)} = \frac{A^-}{T}, \tag{4.12}$$

$$\mathbf{E}(\texttt{Specificity}) = \frac{\mathbf{E}(TN)}{\mathbf{E}(TN) + \mathbf{E}(FP)} = \frac{A^-}{T}. \tag{4.13}$$

Interestingly, expected values for the accuracy measures concerning the identification of positive cases (i.e. `Precision` and `Recall`) are found to be A^+/T, and those relating to negative cases (i.e. `Npv` and `Specificity` are found to be A^-/T. This indicates that, provided that a test data set contains many positive instances (i.e. defect-prone modules), then high `precision` and `recall` are essential, while low `npv` and `specificity` are acceptable.

[6]Specifically, $\mathbf{E}(FP)$, $\mathbf{E}(FN)$ and $\mathbf{E}(TN)$ are respectively A^+A^-/T, A^+A^-/T and $(A^-)^2/T$.

4.3.2 Neg/pos-normalized accuracy measures

To demonstrate how to use expected values in interpreting the values of accuracy measures concerning a certain prediction model, we give a hypothetical example in Figure 4.2. Let M denote an accuracy measure. Suppose that running defect prediction on three data sets A, B, C with a certain defect prediction model yields the accuracy values $\mathtt{m}$, which are demonstrated in Figure 4.2 together with their expected values $\mathbf{E}(\mathtt{M})$.

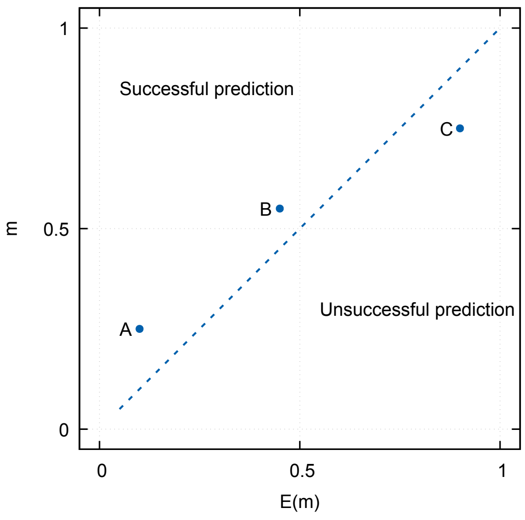


Figure 4.2 Accuracy values $\mathtt{m}$ and expected values $\mathbf{E}(\mathtt{M})$ for three data sets.

If we interpret prediction performance based only on $\mathtt{m}$, the prediction for data set C seems much better than those for A and B. However, note that concerning data set C the expected value of $\mathtt{M}$ is even higher than its empirical value, i.e. $\mathbf{E}(\mathtt{M}) > \mathtt{m}$. In other words, if we keep making random predictions for C, eventually we will get better accuracy than $\mathtt{m}$. In this sense, we consider a prediction to be successful, only if the accuracy measures yield better numbers than their expected values.

In addition to this simple binary (better or worse) assessment, one can also quantify *how much better or worse* the prediction model is than the expected values. Of course, it is plausible to consider that the higher is the empirical value than the expected value, the better is the prediction performance. However, taking the difference between the empirical and expected values straightforwardly is not adequate, since one needs to consider also the variance in possible predictions. In that respect, concerning an accuracy value $\mathtt{m}$, we define the *neg/pos-normalized value* $\bar{\texttt{m}}$ as follows:

$$\bar{\texttt{m}} = \frac{\texttt{m} - \mathbf{E}(\texttt{M})}{\sigma_\pi} \tag{4.14}$$

where σ_π denotes the standard deviation of all possible prediction outcomes. Specifically,

$$\sigma_\pi = \sqrt{\frac{\sum_{i=1}^{\#(\Pi)} (\pi_i - \mathbf{E}(\Pi))^2}{\#(\Pi)}} \tag{4.15}$$

where $\mathbf{E}(\Pi)$ denotes the expected value of π_i (see also Appendix B.1)[7].

Using the neg/pos-normalized measures, we can assess how much better or worse the prediction results are than the expected values. For example, assuming one normal distribution based on all possible predictions, if the neg/pos-normalized `precision`, i.e. $\overline{\texttt{precision}}$, is larger than 0, then the prediction is better than 50% of all possible predictions, and thus it can be considered as successful. On the other hand, if $\overline{\texttt{precision}} > 1$, then the prediction is 1σ better than an average prediction and so it is better than 84% of all possible predictions, which is quite successful.

In this chapter, a prediction for which neg/pos-normalized recall, precision, specificity, NPV are all positive is considered to be a successful prediction, otherwise it is an unsuccessful prediction.

4.4 Case study

The purpose of this case study is to demonstrate that large values of accuracy measures do not necessarily indicate successful predictions and vice versa. For drawing attention to the dangers of using conventional accuracy measures in comparing prediction results across different data sets, we contrast the rankings of data sets in terms of raw and neg/pos normalized accuracy values[8].

[7]Note that in Equation 4.15, $\#(\Pi)$ is simply a shorthand notation for $\#\Big(\Pi(A^-, A^+)\Big)$ appearing in Equation 4.6.

[8]Note that producing a ranking of data sets according to the accuracy values of some prediction model is not one of our purposes. Nevertheless, we consider it as a simple and clear way to demonstrate the discrepancy between raw and neg/pos normalized accuracy values

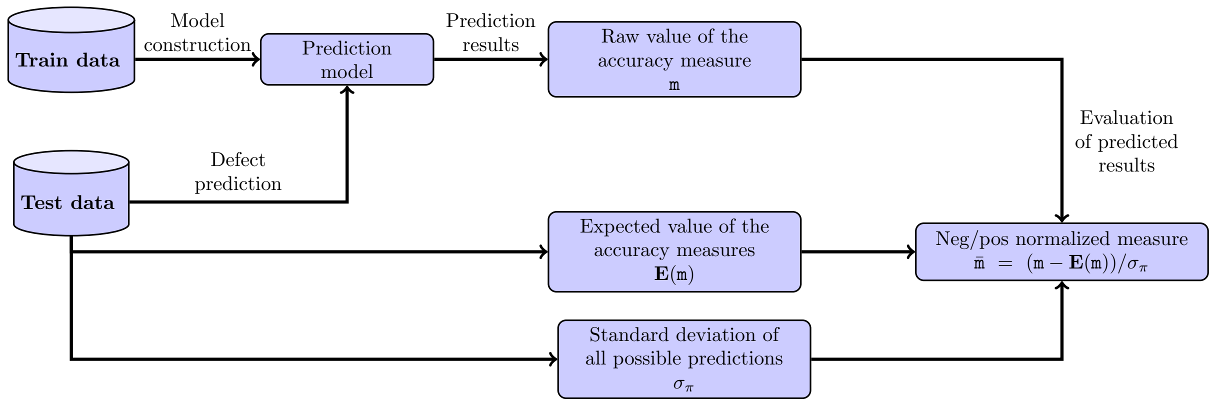

Train data
Model construction
Prediction model
Prediction results
Raw value of the accuracy measure $\mathtt{m}$
Evaluation of predicted results
Test data
Defect prediction
Expected value of the accuracy measures $\mathbf{E}(\mathtt{m})$
Neg/pos normalized measure $\bar{\mathtt{m}} = (\mathtt{m} - \mathbf{E}(\mathtt{m}))/\sigma_\pi$
Standard deviation of all possible predictions σ_π

Figure 4.3 The procedure of cross-version experiment.

4.4.1 Data and methodology

As data, we use the same sets deployed in the preliminary analysis (see Table 4.2). Similar to Section 4.2.2, we conduct a cross-version defect prediction, where the old version of each project is used in training and its new version is used in testing. As a defect prediction model, we employ random forest, since it was shown to be one of the best models in the defect prediction domain and in other classification problems in terms of prediction performance and stability [25, 27, 54, 74, 85].

In addition, we evaluate commonly-used composite measures F1-value, AUC of ROC, MCC, G-mean and Balance by comparing their ranking results with our ranking by neg/pos-normalized measures.

Figure 4.3 shows the procedure of our case study. Firstly, we use training data to conduct model construction. Secondly, we make predictions on the test data and calculate raw values of accuracy measures. Thirdly, we calculate their expected values (e.g. as in Equation 4.10) and the standard deviations (see Equation 4.15). Finally, we calculate corresponding neg/pos-normalized values as in Equation 4.14[9].

In this study, we perform 30 repetitions of the prediction experiment, and take the average of them as the value of the prediction experiment.

[9]While computing raw values of some accuracy measures, there are cases where the denominator becomes zero because of too large neg/pos ratio (e.g. data set no. 19 XDOC). In our experiment, we use $\bar{m} = 0$ for such cases.

Table 4.3: Values of conventional accuracy measures and their neg/pos-normalized values for each data set.

		(a)				(b)				(c)			
		Raw accuracy values				Expected accuracy values				Neg/pos-normalized accuracy values			
No	Project	`prec.`	`recall`	`npv`	`specif.`	**E**(`Prec.`)	**E**(`Recall`)	**E**(`Npv`)	**E**(`Specif.`)	$\overline{\texttt{prec.}}$	$\overline{\texttt{recall}}$	$\overline{\texttt{npv}}$	$\overline{\texttt{specif.}}$
1	MYLN	0.604	0.916	0.913	0.594	0.403	0.403	0.597	0.597	0.497	1.269	0.439	-0.004
2	PDE	0.547	0.257	0.784	0.927	0.256	0.256	0.744	0.744	1.124	0.005	0.051	0.232
3	JDT	0.467	0.325	0.920	0.954	0.109	0.109	0.891	0.891	3.233	1.947	0.033	0.071
4	NBNS	0.462	0.324	0.890	0.936	0.146	0.146	0.854	0.854	2.159	1.211	0.042	0.094
5	ANT	0.606	0.160	0.763	0.963	0.263	0.263	0.737	0.737	1.292	-0.388	0.033	0.288
6	CAML	0.486	0.268	0.836	0.930	0.199	0.199	0.801	0.801	1.430	0.344	0.043	0.156
7	FRST	0.200	0.500	0.963	0.867	0.063	0.063	0.938	0.938	0.788	2.506	0.027	-0.075
8	JEDT	0.128	0.455	0.987	0.929	0.022	0.022	0.978	0.978	2.127	8.738	0.009	-0.050
9	LOG4	0.947	0.282	0.087	0.813	0.922	0.922	0.078	0.078	0.027	-0.694	0.010	0.794
10	LUCN	0.651	0.691	0.496	0.451	0.597	0.597	0.403	0.403	0.090	0.157	0.130	0.067
11	POI	0.746	0.721	0.537	0.568	0.637	0.637	0.363	0.363	0.170	0.132	0.238	0.279
12	PROP	0.552	0.028	0.850	0.996	0.153	0.153	0.847	0.847	2.611	-0.813	0.004	0.173
13	SYNP	0.619	0.106	0.747	0.976	0.270	0.270	0.730	0.730	1.270	-0.600	0.022	0.317
14	ECOS	0.080	0.320	0.986	0.927	0.019	0.019	0.981	0.981	2.390	11.778	0.005	-0.054
15	EXIM	0.895	0.287	0.567	0.966	0.508	0.508	0.492	0.492	0.755	-0.432	0.107	0.670
16	GNY	0.456	0.419	0.721	0.751	0.333	0.333	0.667	0.667	0.361	0.251	0.073	0.113
17	HLMA	0.236	0.335	0.850	0.775	0.170	0.170	0.830	0.830	0.348	0.873	0.024	-0.065
18	HBNT	0.705	0.019	0.761	0.997	0.241	0.241	0.759	0.759	1.916	-0.922	0.004	0.300
19	XDOC	0.000	0.000	0.706	1.000	0.294	0.294	0.706	0.706	-0.973	-0.973	0.000	0.385

4.4.2 Comparison of accuracy measures to their expected values

Figure 4.4 and Table 4.3 show the relationship between raw (empirical) values of the four conventional accuracy measures as well as their expected and neg/pos normalized values[10]. Figure 4.4 provides the overall distribution of raw and expected values and Table 4.3 lists the values accurately.

Regarding positive accuracy measures, we first examine `Precision` in Figure 4.4-(a) and `Recall` in Figure 4.4-(b)[11].

[10]In Figure 4.4, each color denotes a different data set. The gradation of colors does not imply any progressive relationship between data sets.

[11]In Figure 4.4-(a), there is one unsuccessful prediction. Namely, for data set no.19 (XDOC) `Precision` $=0$, since a valid prediction model could not be constructed due to too few instances of positive cases in training data (see also Table 4.2).

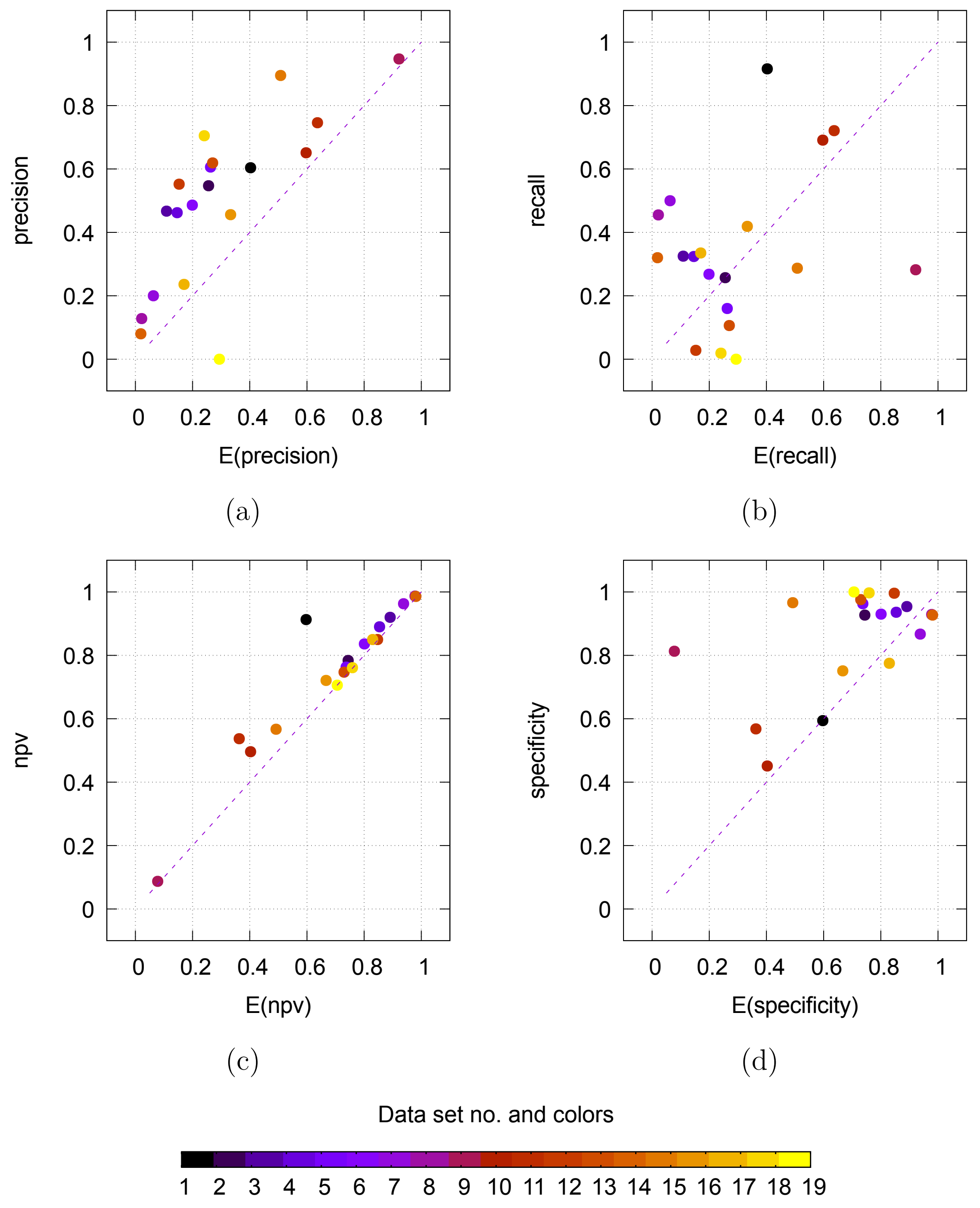


Figure 4.4: The scatter diagram of raw values and expected values. (a) `Precision`, (b) `Recall`, (c) `Npv` and (d) `Specificity`.

In Figure 4.4-(a), the data point at the top right corner represents data set no.9 (LOG4), which attains a remarkably high `precision`. Nevertheless, it is almost the same as its expected value and the neg/pos ratio is considerably low (i.e. 0.085, see also Table 4.2) [12]. Thus, although the `precision` is high, it is not sufficient to prove the efficacy of the prediction model. Therefore, we check the `recall` value relating to this data set in Figure 4.4-(b) (see the right most data point) and see that the raw value is lower than the expected value. We can also confirm In Table 4.3-(b) that $\texttt{recall} = 0.282$ and $\mathbf{E}(\texttt{Recall}) = 0.922$, which is a significant difference. Hence, although we cannot draw a reliable inference from `precision`, based on `recall`, we can say that this prediction is not good. As a matter of fact, since this data set has many defect-prone modules (see Table 4.2), there is little value in predicting defects, and it is preferable to test just all modules.

On the other hand, data set no.14 (ECOS) attains the lowest `precision` and yet it is more than three times larger than its expected value (i.e. $\texttt{precision} = 0.080$, $\mathbf{E}(\texttt{Precision}) = 0.019$, see Table 4.3-(a), (b)). Note that relating to this data set also the `recall` is much greater than its expected value (i.e. $\texttt{recall} = 0.320$, $\mathbf{E}(\texttt{Recall}) = 0.019$, see Table 4.3-(a), (b)) and also that the neg/pos ratio is considerably high (i.e. 50.6, see Table 4.2). Therefore, for data set no.14 the low `precision` is considered not to imply a bad prediction. Similarly, despite fair values of `recall`, the predictions relating to data sets no.1 (MYLN) and no.14 (ECOS) are actually successful, since their `recall` values (and other accuracy measures) are greater than their expected values.

Regarding negative accuracy measures, we examine `npv` in Figure 4.4-(c) and `specificity` in Figure 4.4-(d).

In Figure 4.4-(c), `npv` of all data sets are seen to be very close to their expected values. This indicates that a small/large `npv` implies a small/large expected value $\mathbf{E}(\texttt{Npv})$, which makes it insufficient for the judgment of in/efficiency of prediction performance.

Therefore, we examine the `specificity` values given in Figure 4.4-(d). Five data sets (no. 1, 7, 8, 14, 17) are seen to have lower `specificity` than their expected values (see also Table 4.3). For example, data set no. 8 (JEDT) has $\texttt{specificity} = 0.929$, which is very high but not as high as its expected value $\mathbf{E}(\texttt{Specificity}) = 0.978$. Thus, despite the high `specificity`, we cannot say that the prediction is successful. Note also that the relating neg/pos ratio is 43.727 (see Table 4.2). On the other hand, data set no. 10

[12]Actually, we can read the exact values relating to data set no.9 as $\texttt{precision} = 0.947$ and $\mathbf{E}(\texttt{Precision}) = 0.922$) in Table 4.3-(a) and (b).

(LUCN) has the lowest `specificity` of 0.451, but it is greater than its expected value $\mathbf{E}(\texttt{Specificity}) = 0.403$.

These results indicate the danger of linking the empirical value of (raw) accuracy measures directly to performance evaluation, and also show that misleading inferences can be avoided by the proposed comparison procedure with corresponding expected values.

Note that the probability of detection (*pd*) and the probability of false alarm (*pf*), which are often used in defect prediction studies [66, 65], are considered to be unreliable in some datasets. It is because *pd* and *pf* are referred as "`recall`" and "`1-specificity`" in this study and neg/pos-normalized `recall` and `specificity` are not always reliable as shown in Table 4.3.

4.4.3 Performance evaluation with neg/pos-normalized measures

In this section, we first compute the neg/pos-normalized measures (see Equation 4.14) corresponding to the four conventional accuracy measures given in Equations 4.1∼4.4. Then, we provide two performance rankings with respect to the raw and neg/pos normalized values of the accuracy measures

Ranking of predictions for different data sets

Table 4.3-(a) and (c) show respectively the raw and neg/pos-normalized values concerning the four conventional accuracy measures.

As it can be seen in Table 4.3-(c), for several data sets the neg/pos normalized values are negative, especially for $\overline{\texttt{recall}}$ and $\overline{\texttt{specificity}}$. Notably, data set no. 3 (JDT) has the highest $\overline{\texttt{precision}}$ of 3.233, although the `precision` itself is not very high (i.e. 0.467). On the other hand, data set no. 9 (LOG4) has the lowest $\overline{\texttt{precision}}$ of 0.027, although the `precision` itself is the highest (i.e. 0.947). Thus, it is already evident from these tables that neg/pos normalization leads to a large difference in relative performances.

Subsequently, we rank the predictions with respect to their performance in terms of all accuracy measures. To that end, we use the win-tie-loss method [45, 77] and we aggregate all pairwise win-tie-loss values concerning the 4 accuracy measures into one because they are based on similar principles. Specifically, concerning a particular accuracy measure, if data set A attains a higher value than data set B, the number of *win*s concerning A and the number of *losses* concerning B are incremented by one, whereas for equal values, the number of *tie*s concerning A and B is added by one[13].

Table 4.4: Ranking of predictions for 19 data sets based on (a) conventional (raw) accuracy measures and (b) neg/pos-normalized accuracy measures.

(a)						(b)					
	Rank	Win	Tie	Loss	Win-Loss		Rank	Win	Tie	Loss	Win-Loss
MYLN	1	45	0	27	18	JDT	1	48	1	23	25
JDT	1	45	0	27	18	NBNS	2	46	0	26	20
JEDT	3	43	0	29	14	EXIM	3	44	0	28	16
EXIM	4	42	0	30	12	CAML	3	44	0	28	16
FRST	5	40	0	32	8	MYLN	5	42	0	30	12
HBNT	5	40	0	32	8	PDE	6	40	0	32	8
NBNS	5	40	0	32	8	POI	6	40	0	32	8
PROP	8	39	1	32	7	ANT	8	39	1	32	7
SYNP	9	37	0	35	2	ECOS	9	39	0	33	6
ANT	9	37	0	35	2	JEDT	10	38	0	34	4
POI	11	36	0	36	0	GNY	11	37	0	35	2
ECOS	12	34	1	37	-3	SYNP	12	35	0	37	-2
CAML	13	34	0	38	-4	FRST	13	32	0	40	-8
HLMA	14	31	1	40	-9	LUCN	13	32	0	40	-8
LUCN	15	31	0	41	-10	PROP	15	30	1	41	-11
PDE	16	30	1	41	-11	HBNT	16	29	1	42	-13
LOG4	17	30	0	42	-12	LOG4	17	27	0	45	-18
GNY	18	26	0	46	-20	HLMA	18	24	0	48	-24
XDOC	19	22	0	50	-28	XDOC	19	16	0	56	-40

Table 4.4-(a) shows the ranking of data sets based on conventional (raw) accuracy measures, whereas Table 4.4-(b) shows the ranking based on the neg/pos-normalized values of those measures. We immediately notice that the two rankings are quite different. Some of the data sets, for which neg/pos normalization leads to a large change in rank, involve CAML ($13 \rightarrow 3$), GNY ($18 \rightarrow 11$), and JEDT ($3 \rightarrow 10$). Taking a closer look at CAML data set, we see that it ranks 11, 13, 9, and 9 with respect to `precision`, `recall`, `npv`,

[13]Note that the original win-tie-loss method uses a statistical test to judge win or loss, whereas this chapter simply judges based on the difference in accuracy values, since statistical tests (such as Wilcoxon rank-sum test) do not apply to accuracy measures in two-group classification.

`specificity`, respectively. And, after neg/pos normalization the respective ranks become 7, 8, 7, 10, rising a few degrees on average.

Table 4.5 Successful prediction and (raw) value of composite measures.

	Successful or not	F1-value	AUC of ROC	MCC	G-mean	Balance
MYLN	NO	0.728	0.857	0.513	0.738	0.707
PDE	YES	0.350	0.671	0.247	0.488	0.472
JDT	YES	0.383	0.749	0.328	0.557	0.521
NBNS	YES	0.381	0.755	0.302	0.550	0.519
ANT	NO	0.253	0.766	0.213	0.392	0.405
CAML	YES	0.345	0.669	0.252	0.499	0.480
FRST	NO	0.286	0.930	0.244	0.658	0.634
JEDT	NO	0.199	0.730	0.209	0.650	0.611
LOG4	NO	0.435	0.551	0.057	0.479	0.476
LUCN	YES	0.670	0.639	0.145	0.558	0.555
POI	YES	0.733	0.659	0.286	0.640	0.636
PROP	NO	0.054	0.619	0.099	0.168	0.313
SYNP	NO	0.180	0.658	0.172	0.320	0.367
ECOS	NO	0.128	0.763	0.128	0.544	0.517
EXIM	NO	0.434	0.571	0.342	0.526	0.495
GNY	YES	0.437	0.657	0.173	0.560	0.553
HLMA	NO	0.276	0.643	0.097	0.508	0.503
HBNT	NO	0.036	0.629	0.087	0.137	0.306
XDOC	NO	0	0.654	0	0	0.293

Comparison with conventional composite measures

As composite accuracy measures, we use the F1-value, AUC of ROC, MCC, G-mean and Balance since these are commonly used accuracy measures in two-group classification studies (including software defect prediction). We compute their (raw) values (we perform 30 repetitions and take their average as shown in Table 4.5), rank them according to these values and compare this ranking to the one reported in Table 4.4.

In addition, we evaluate the parallel between each pair of rankings based on the correlation coefficient (r). Note that a correlation coefficient (in absolute value) smaller than 0.36 is

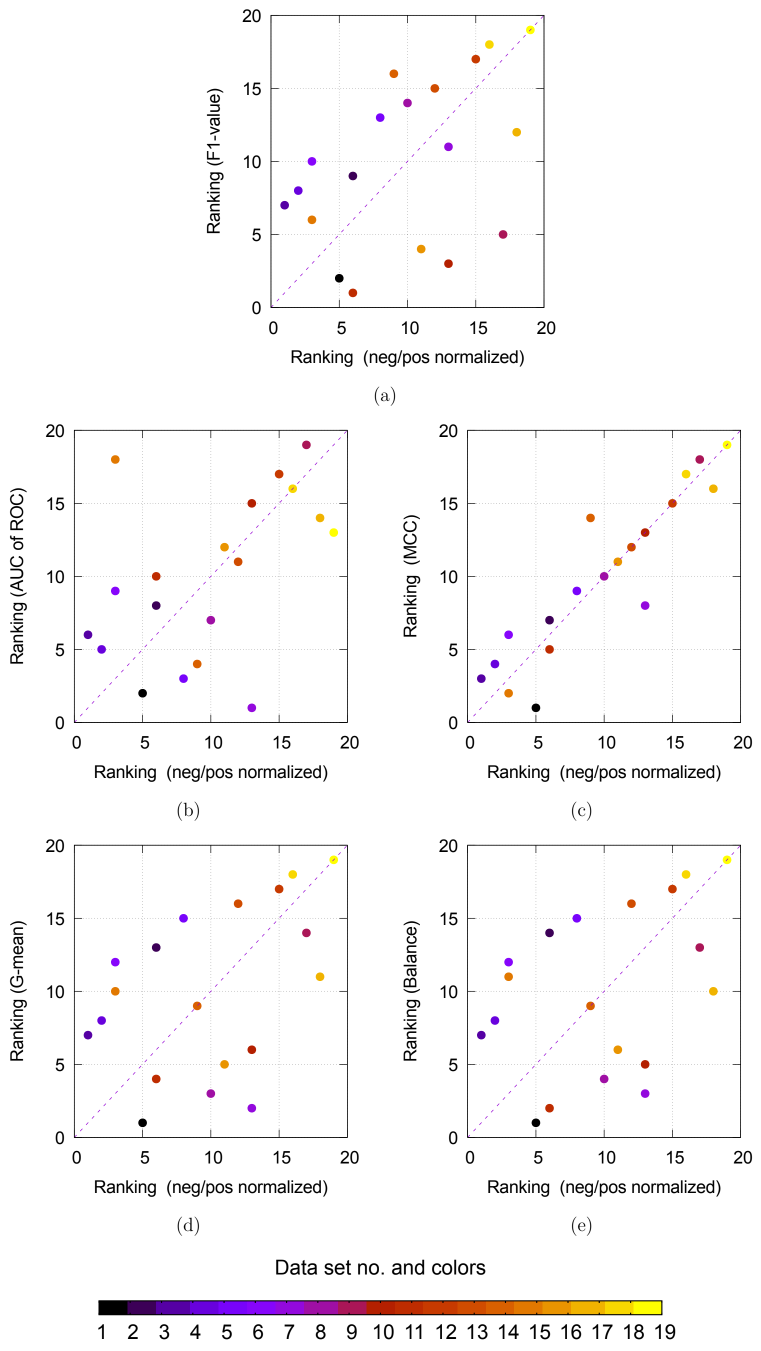


Figure 4.5: The relationship between the rankings based on (a) F1-value, (b) AUC of ROC, (c) MCC, (d) G-mean, (e) Balance and the ranking based on neg/pos normalized accuracy measures (see Table 4.4-(b)). Correlations coefficients are 0.469, 0.486, 0.920, 0.430 and 0.401, respectively.

generally considered to indicate a low or weak correlation, 0.36 to 0.68 a modest or moderate correlation, and 0.68 to 1.0 a strong or high correlation [87].

Figure 4.5 shows the distribution of pairs of rankings for each composite accuracy measure. In each diagram, the x-axis represents the ranking based on neg/pos-normalized measure (given in Table 4.4-(b)) and the y-axis represents the ranking based on (raw value of) the composite measure.

Based on the values presented in Figure 4.5, we find the correlation between MCC and its neg/pos-normalized version to be quite strong ($r = 0.920$). On the other hand, the correlation of AUC of ROC, F1-value, G-mean and Balance with their respective neg/pos-normalized versions to be moderate (i.e. $r = 0.486$, $r = 0.469$, $r = 0.430$ and $r = 0.401$, respectively). Therefore, we recommend using MCC rather than the F1-value, AUC of ROC, G-mean and Balance.

Regarding the identification of successful/unsuccessful prediction, the composite measures are sometimes not useful. For example, data set no. 1 (MYLN) has high F1-value (0.728), AUC of ROC (0.857), MCC (0.513), G-mean (0.738) and Balance (0.707) as shown in Table 4.5, but according to the Table 4.3, its neg/pos-normalized recall, precision, specificity, NPV are not all positive, therefore the prediction for the data set is considered to be unsuccessful. On the other hand, data set no. 6 (CAML) has lower F1-value (0.345), AUC of ROC (0.669), MCC (0.252), G-mean (0.499) and Balance (0.480), but its neg/pos-normalized recall, precision, specificity, NPV are all positive, so the prediction for this data set is considered to be successful.

4.5 Threats to Validity

We provide a discussion on the validity of the proposed method in terms of three commonly adopted experimental validation approaches, i.e. internal validity, external validity and construct validity.

Internal validity refers to the extent to which the observed effect is a consequence of the presumed cause. In our case, one possible issues of internal validity are worth mentioning. This issue is that we used a single prediction method. In that respect, our important future work is to employ other prediction methods such as support vector machines and logistic regression models to increase the validity of the result.

External validity refers to the generalization of the results. In this study, we address external validity by using 38 releases of 19 open-source software project data sets with diverse characteristics obtained from different sources. Namely, they vary in number (i.e. the number of bugs), and project variables, as well as origin (i.e. recording organization) and recording period. In addition, we applied our method to four different accuracy measures Precision, Recall, NPV and Specificity. Our future work is to employ more data sets and accuracy measures to increase the generalization of the results.

Construct validity refers to the relevance and capability of the observations and measurements in evaluating the posed hypothesis. In this study, we compare the results of the proposed normalization with the (raw) values of conventional measures. It is our future work to employ other methods (e.g. human evaluation) to increase the validity of our work.

4.6 Conclusions and future prospects

This chapter describes a way to compute *expected values* of accuracy measures based on all possible prediction outcomes. Based on the expected values, we define *neg/pos-normalized accuracy measures* as the difference between the actual and expected values divided by the standard deviation of all possible prediction outcomes. A case study of defect prediction with 19 data sets show that even a low accuracy value (e.g. $\texttt{precision} < 0.1$) could be considered to indicate a successful prediction (e.g. the case of data set no.14 (ECOS)), and a high accuracy value (e.g. $\texttt{precision} > 0.8$) can imply failure (e.g. the case of data set no.9 (LOG4) because it is too close to the baseline), depending on the composition (i.e. neg/pos ratio) of the data set. We also compare our ranking based on the neg/pos-normalized measures with conventional ranking using F1-value, AUC of ROC, MCC, G-mean and Balance, and found that MCC showed the highest correlation ($r = 0.920$) with our ranking; thus, we recommend using MCC rather than F1-value, AUC of ROC, G-mean and Balance.

As future work, we will employ other prediction models with more data sets and accuracy measures to increase the generalization of the results and to increase the construct validity of our work.

Chapter 5

Conclusions

In this paper we proposed data generation and evaluation methods for mining software engineering data sets that can support data gathering, preparation and evaluation. In the Chapter 2, we proposed a method for artificially generating a mimic software project data set to preserve the confidentiality of data sets. Our experiments confirmed that models built from mimic data sets show similar effort estimation performance as the models built from original data sets, which indicate the capability of the proposed method in generating representative samples.

In the Chapter 3, we proposed a data quality SCIL, that can be used before applying data mining. Our empirical evaluation showed that SCIL can distinguish between consistent and inconsistent data sets. also the experiment showed that prediction models for software development effort and productivity built from consistent data sets achieved a relatively higher accuracy.

In the Chapter 4, defect prediction performance measures called neg/pos-normalized accuracy measures. Our case study of defect prediction indicate that ranking of predictions is significantly different than the ranking of conventional accuracy measures such as precision and recall as well as composite measures F1-value, AUC of ROC, MCC, G-mean and Balance. In addition, we conclude that MCC attains a better defect prediction accuracy than F1-value, AUC of ROC, G-mean and Balance.

One of the important future work is to employ more data sets to evaluate the proposed methods to increase the generalization of the results and to increase the construct validity of our work. Another important future work is to explore methods to support the remaining data mining steps, *i.e.* setting objectives and applying data mining algorithms. In particular, hypothesis building in the setting objectives step is often very difficult because hypotheses can be constructed after data mining is applied. Therefore, methods to support the continuous loop between hypothesis construction and data mining are an important target for future research.

Appendix A

A.1 Distance Metrics

We denote Euclidean distance between projects $\mathbf{p}_i$ and $\mathbf{p}_j$ with $d_E(\mathbf{p}_{ij})$, where

$$d_E(\mathbf{p}_{ij}) = \sqrt{\sum_{f_m} (\mathbf{p}_i[f_m] - \mathbf{p}_j[f_m])^2}. \tag{.1}$$

In addition, we denote the cosine distance between the same pair with $d_C(\mathbf{p}_{ij})$, where

$$d_C(\mathbf{p}_{ij}) = \frac{\sum_{f_m} \mathbf{p}_i[f_m] \cdot \mathbf{p}_j[f_m]}{\sqrt{\sum_{f_m} \mathbf{p}_i^2[f_m]} + \sqrt{\sum_{f_m} \mathbf{p}_j^2[f_m]}}. \tag{.2}$$

A.2 Normalizations

Let $\mathbf{p}_i$ be an arbitrary project from a data set D, f_m be an arbitrary feature and $\bar{\mathbf{p}}_\mathbf{i}[f_m]$ be the MinMax normalized value of that feature relating project $\mathbf{p}_i$.

$$\bar{\mathbf{p}}_\mathbf{i}[f_m] = \frac{\mathbf{p}_\mathbf{i}[f_m] - f_{m,min}}{f_{m,max} - f_{m,min}} \tag{.3}$$

where $f_{m,min}$ and $f_{m,max}$ are the minimum and maximum values of that feature over all projects in the data set.

$$f_{m,min} = \min_{\mathbf{p}_\mathbf{i} \in D} (\mathbf{p}_i[f_m])$$

$$f_{m,max} = \max_{\mathbf{p}_\mathbf{i} \in D} (\mathbf{p}_i[f_m])$$

Let $\bar{\bar{\mathbf{p}}}_\mathbf{i}[f_m]$ be the z-normalized value of the feature f_m relating project $\mathbf{p}_i$.

$$\bar{\bar{\mathbf{p}}}_\mathbf{i}[f_m] = \frac{\mathbf{p}_\mathbf{i}[f_m] - \mu_i}{\sigma_i} \tag{.4}$$

where μ_i is the mean value and σ_i is the standard deviation of that feature over projects in the data set.

A.3 The effect of pre-processing and estimation method on the performance of CIL

In this section, we assess the improvement on CIL that can be expected by applying pre-processing operations and by improving the estimator performance.

As the target variable, we focus on effort. As pre-processing, we employ Euclidean distance d_E coupled with MinMax normalization and weighting, since this combination is determined to be the best for CIL in Section 3.6.6. As for estimator model, we use first CART + Tree pruning [53], which is claimed to be the most efficient estimator by Phannachitta *et al.* [78], and then Random Forest, which is demonstrated empirically to perform better than CART + Tree pruning in Section 3.6.2.

Similar to Table 3.3, Table A.1 presents the correlation between CIL and MMRE for the target variable of effort with the estimator model of CART + Tree pruning. However, unlike Table 3.3 the input data is pre-processed in Table A.1. Since the absence/presence of pre-processing is the only difference, we can make a direct comparison between Tables 3.3 and A.1 to assess the effect induced on CIL by pre-processing.

By examining these tables, we observe that R values are higher in Table A.1 in most but not all cases. This indicates that the lack of a strong correlation between CIL and MMRE is partially due to the lack of pre-processing. Nevertheless, it can not be attributed solely

Table A.1: The correlation coefficients R concerning **MMRE** and **CIL** values in the follow-up evaluation (with pre-processing and CART + Tree pruning).

Data set	R		
	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$
China [70]	0.324	0.388	0.476
Coc81dem [6]	-0.435	-0.305	-0.106
Desharnais [20]	-0.592	-0.260	0.265
Maxwell [64]	-0.380	-0.277	-0.261
Miyazaki94 [72]	-0.230	-0.586	-0.593
Nasa93 [70]	0.196	0.382	0.459

Table A.2: The correlation coefficients R concerning **MMRE** and **CIL** values in the follow-up evaluation (with pre-processing and Random Forest).

Data set	R		
	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.5$
China [70]	0.034	-0.130	-0.052
Coc81dem [6]	-0.242	-0.111	0.060
Desharnais [20]	-0.413	-0.005	0.608
Maxwell [64]	-0.395	-0.208	-0.357
Miyazaki94 [72]	-0.492	-0.227	-0.137
Nasa93 [70]	-0.277	0.102	0.208

to that. In addition, the positive values in Table A.1 are quite small, and thus, CIL is very far from satisfactory even with the most efficient pre-processing combination (among those addressed in this study).

Similar to Table A.1, Table A.2 presents the correlation between CIL and MMRE for the target variable of effort and with pre-processing. However, in Table 3.3 the estimator model is CART + tree pruning and in Table A.1 it is Random Forest. Since the estimator model is the only difference, we can make a direct comparison between Tables A.1 and A.2 to assess the effect induced on CIL by improvement of the estimator model.

Note that in Table A.2, we support CIL not only by applying the pre-processing operations but also by integrating it with a better estimator model (i.e. replacing CART + Tree pruning with Random Forest). In that respect, Table A.2, gives an insight to the maximum improvement that we can expect on CIL by applying the best execution mode identified in this study.

However, comparing Tables A.1 and A.2, we see that improving the estimator does not necessarily lead to an increase in the correlation between CIL and MMRE. By comparing corresponding values in these tables, it is seen that there are more cases where R degrades than where it improves. Thus, we conclude that the low correlation between MMRE and CIL cannot be blamed on the poor performance of the estimation model either.

A.4 Results concerning alternative threshold values

Figures A.1 $\sim$ A.3 show scatter diagrams of SCIL and MMRE of effort estimation for different threshold α values (0.1, 0.3 and 0.5). Figures A.4 $\sim$ A.6 show scatter diagrams of SCIL and MMRE of productivity estimation for different threshold α values (0.1, 0.3 and 0.5).

Figures A.7 $\sim$ A.9 show scatter diagrams of CIL and MMRE of effort estimation for different threshold α values (0.1, 0.3 and 0.5). Figures A.10 $\sim$ A.12 show scatter diagrams of CIL and MMRE of productivity estimation for different threshold α values (0.1, 0.3 and 0.5).

In each diagram, the title shows the distance function used, normalization technique used, and weighting used or not (e.g. Cosine-MinMax-noWT means the cosine distance and Min-Max normalization without weighting). Note that for IVDM distance function, we did not apply normalization and weighting because it already considers the relationship between the target variable and feature variables in distance computation.

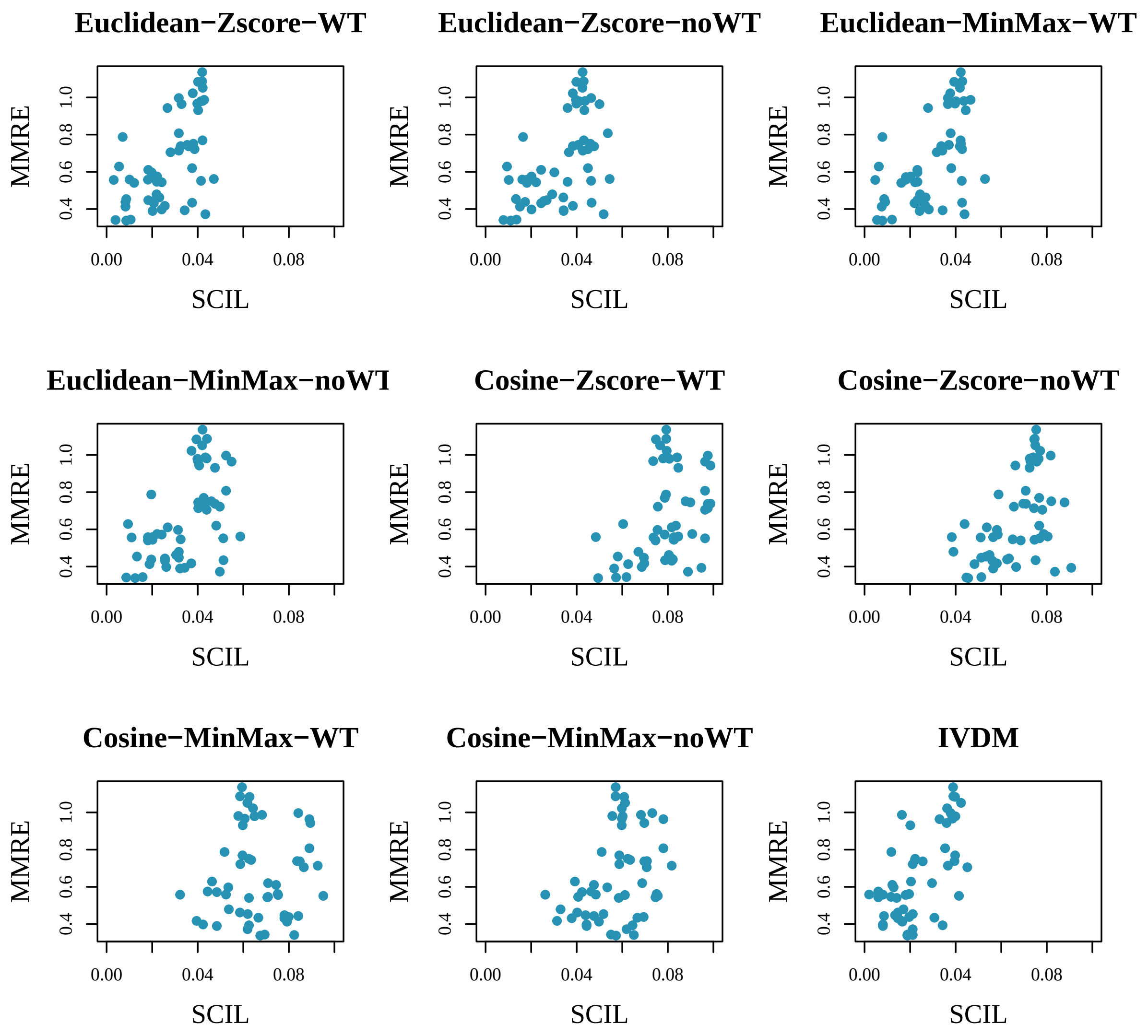


Figure A.1: The scatter plot of SCIL vs MMRE for the estimation target of effort concerning Euclidean distance d_E, Cosine distance d_C and IVDM. Note that the threshold α is set to 0.1 for all plots.

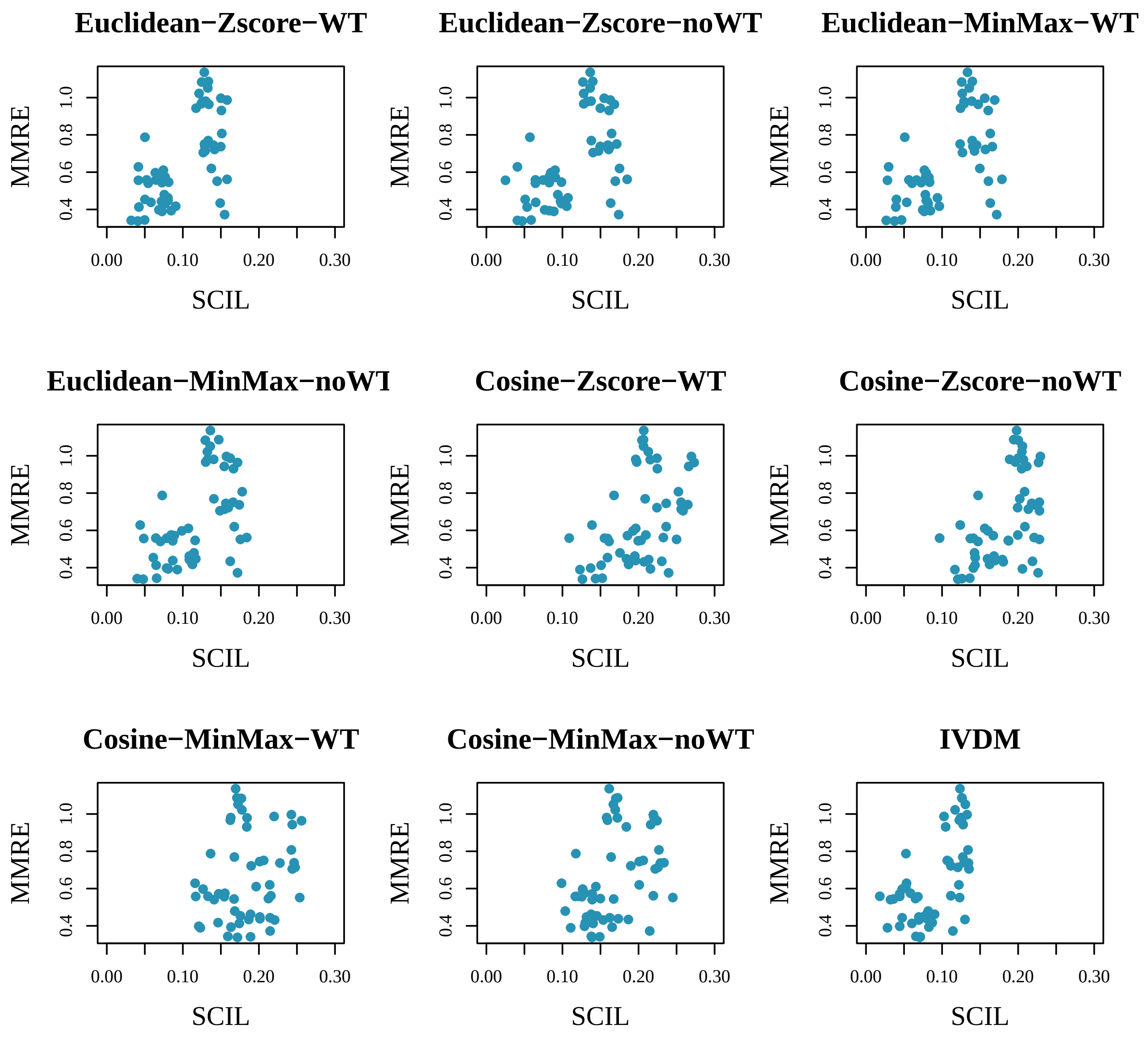


Figure A.2: The scatter plot of SCIL vs MMRE for the estimation target of effort concerning Euclidean distance d_E, Cosine distance d_C and IVDM. Note that the threshold α is set to 0.3 for all plots.

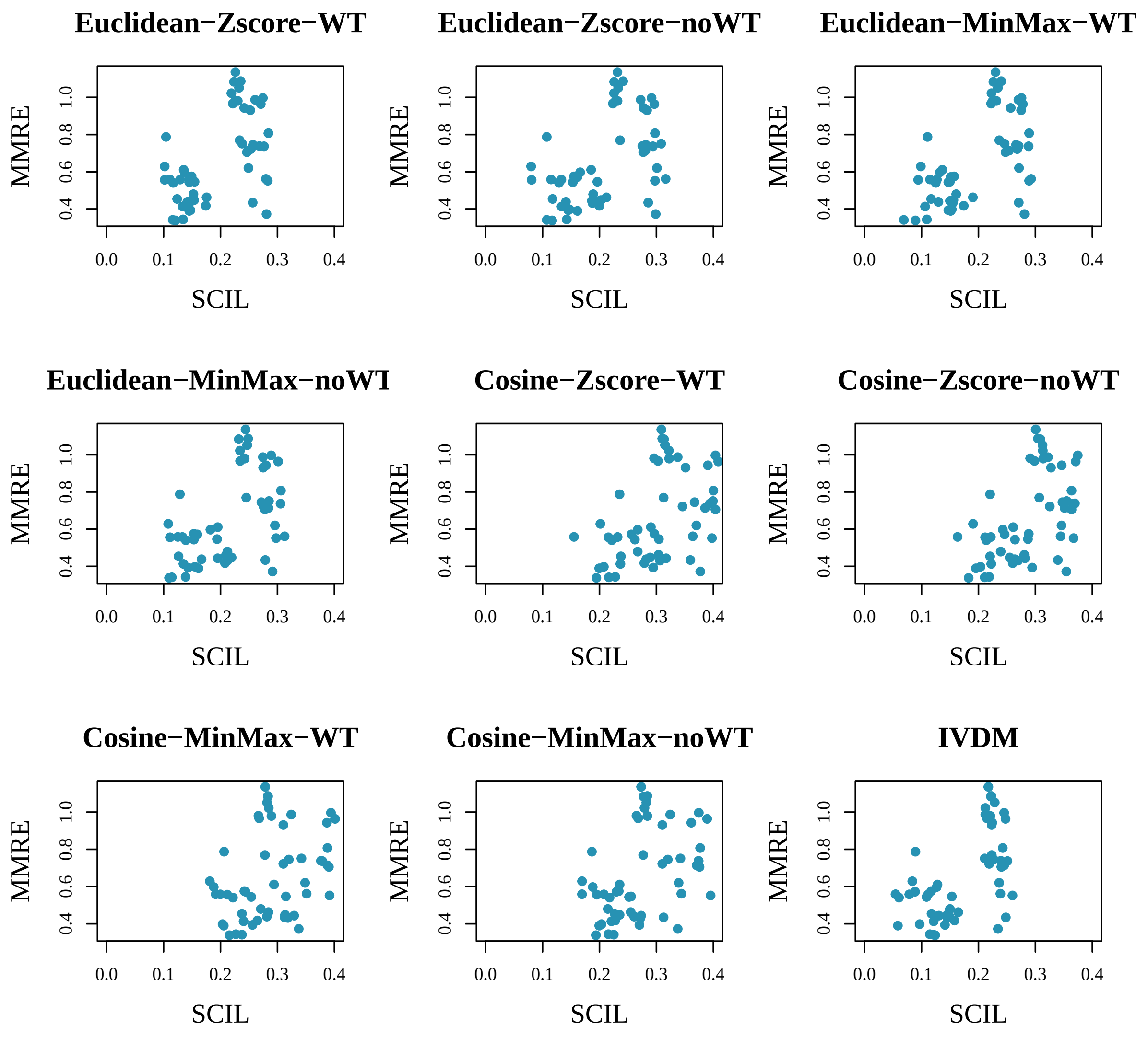


Figure A.3: The scatter plot of SCIL vs MMRE for the estimation target of effort concerning Euclidean distance d_E, Cosine distance d_C and IVDM. Note that the threshold α is set to 0.5 for all plots.

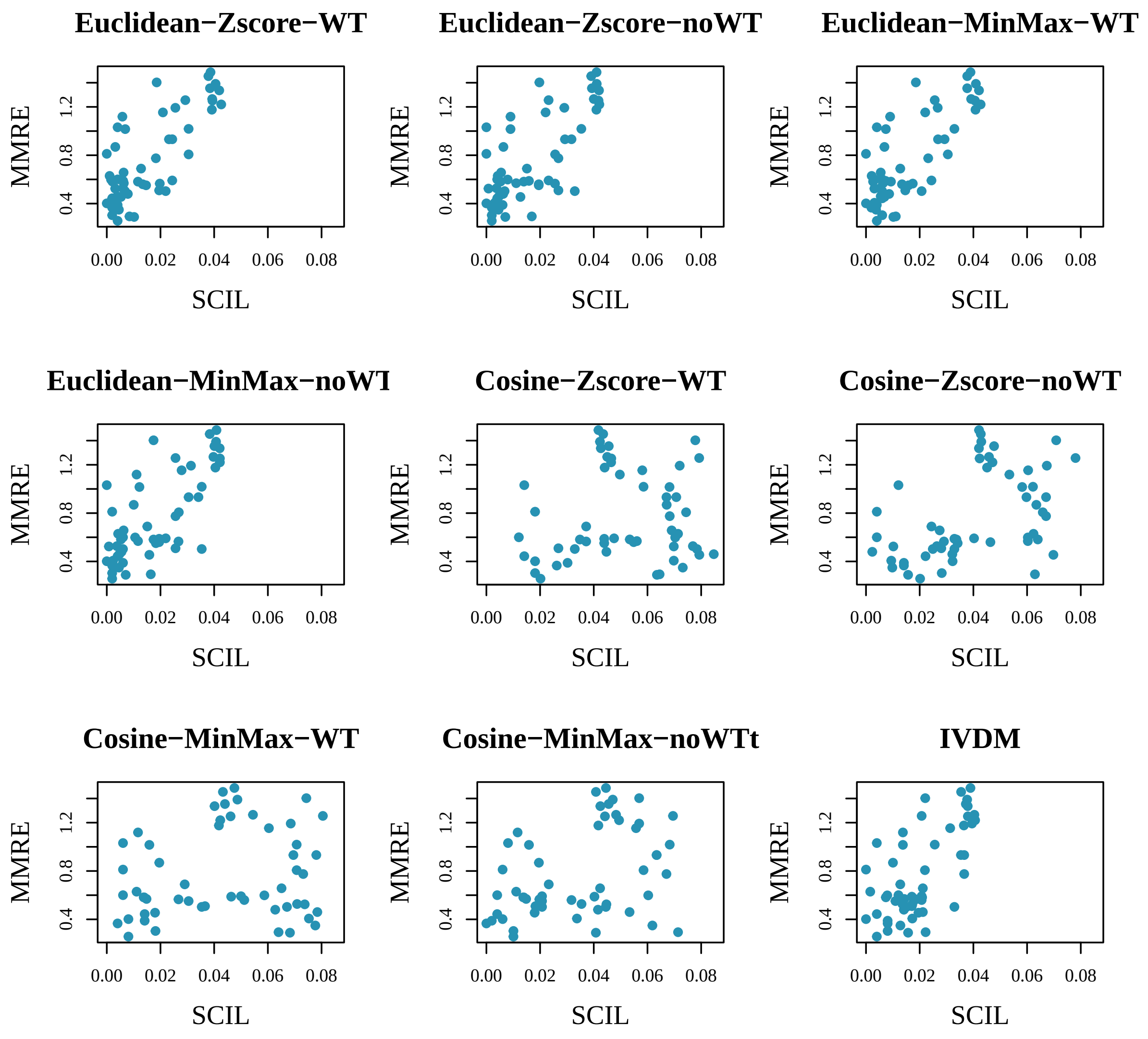


Figure A.4: The scatter plot of SCIL vs MMRE for the estimation target of productivity concerning Euclidean distance d_E, Cosine distance d_C and IVDM. Note that the threshold α is set to 0.1 for all plots.

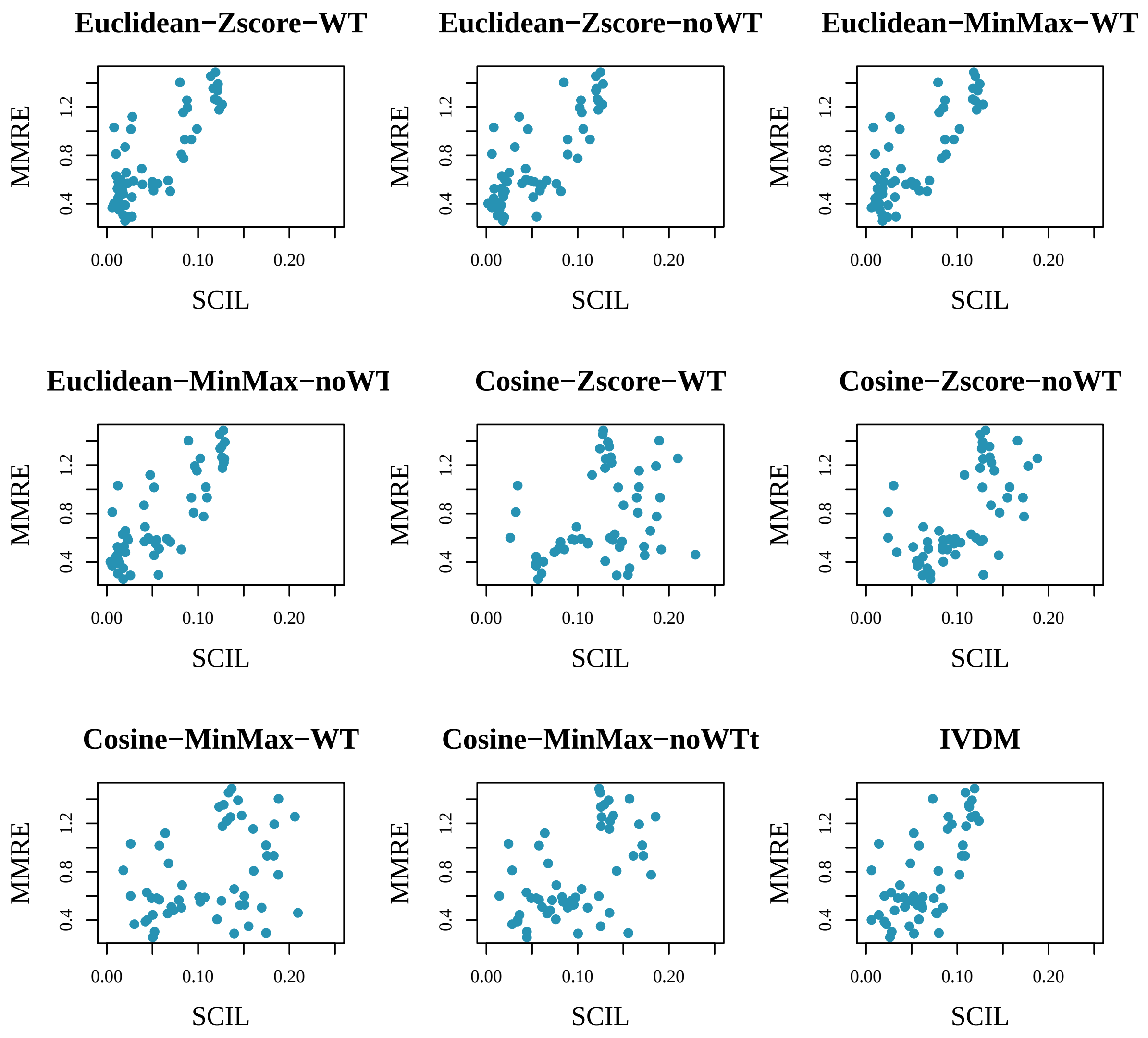


Figure A.5: The scatter plot of SCIL vs MMRE for the estimation target of productivity concerning Euclidean distance d_E, Cosine distance d_C and IVDM. Note that the threshold α is set to 0.3 for all plots.

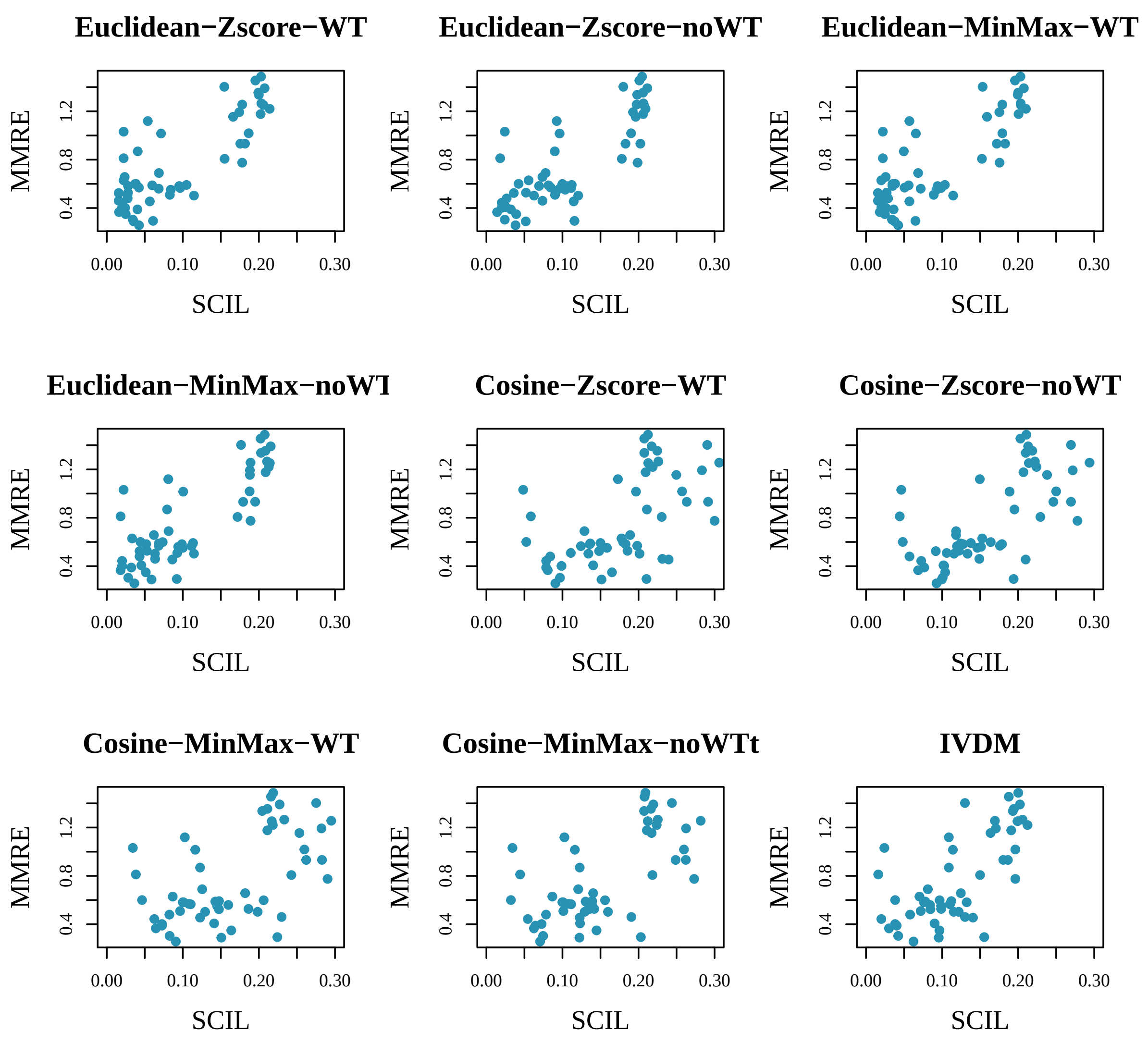


Figure A.6: The scatter plot of SCIL vs MMRE for the estimation target of productivity concerning Euclidean distance d_E, Cosine distance d_C and IVDM. Note that the threshold α is set to 0.5 for all plots.

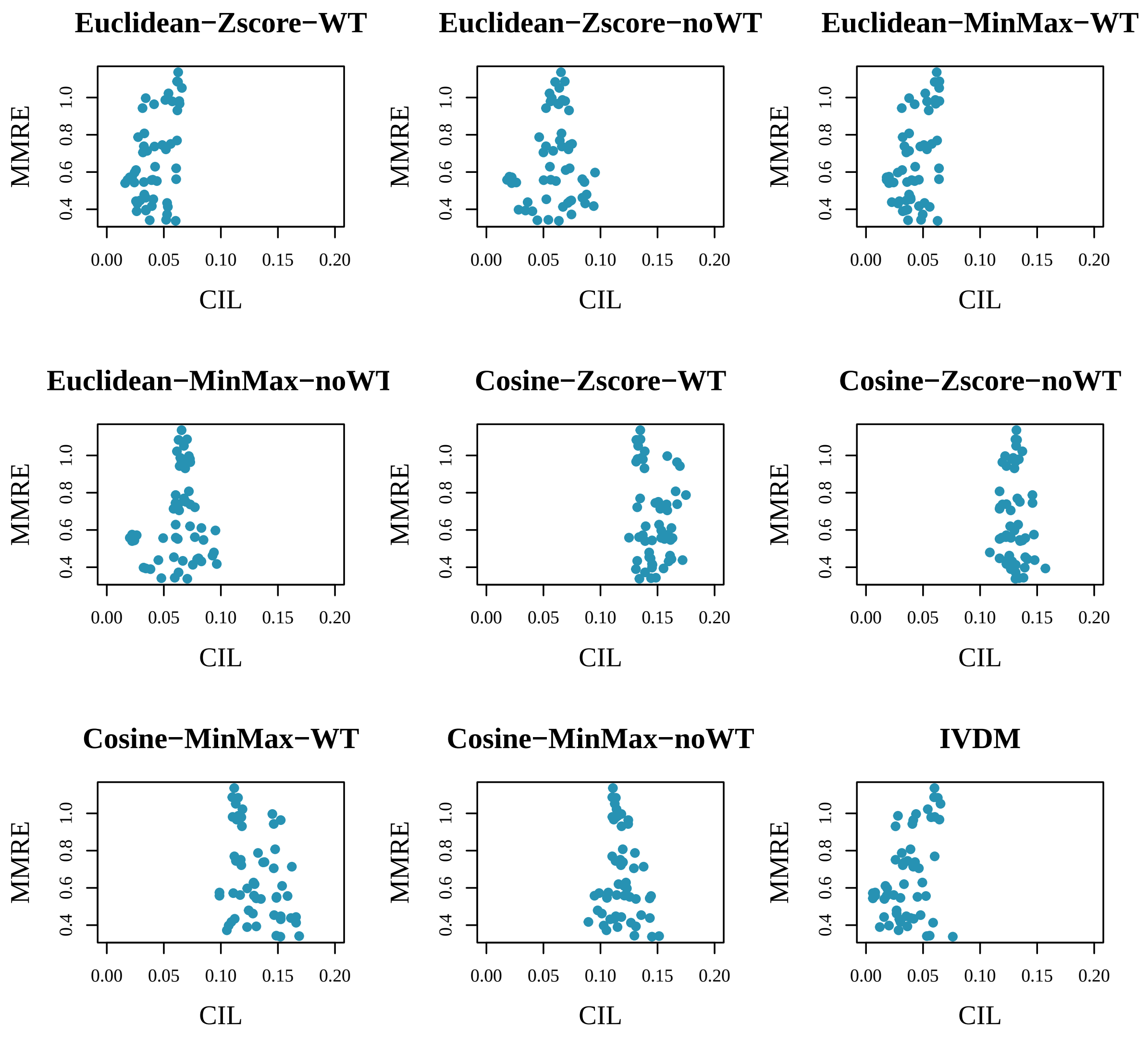


Figure A.7: The scatter plot of CIL vs MMRE for the estimation target of effort concerning Euclidean distance d_E, Cosine distance d_C and IVDM. Note that the threshold α is set to 0.1 for all plots.

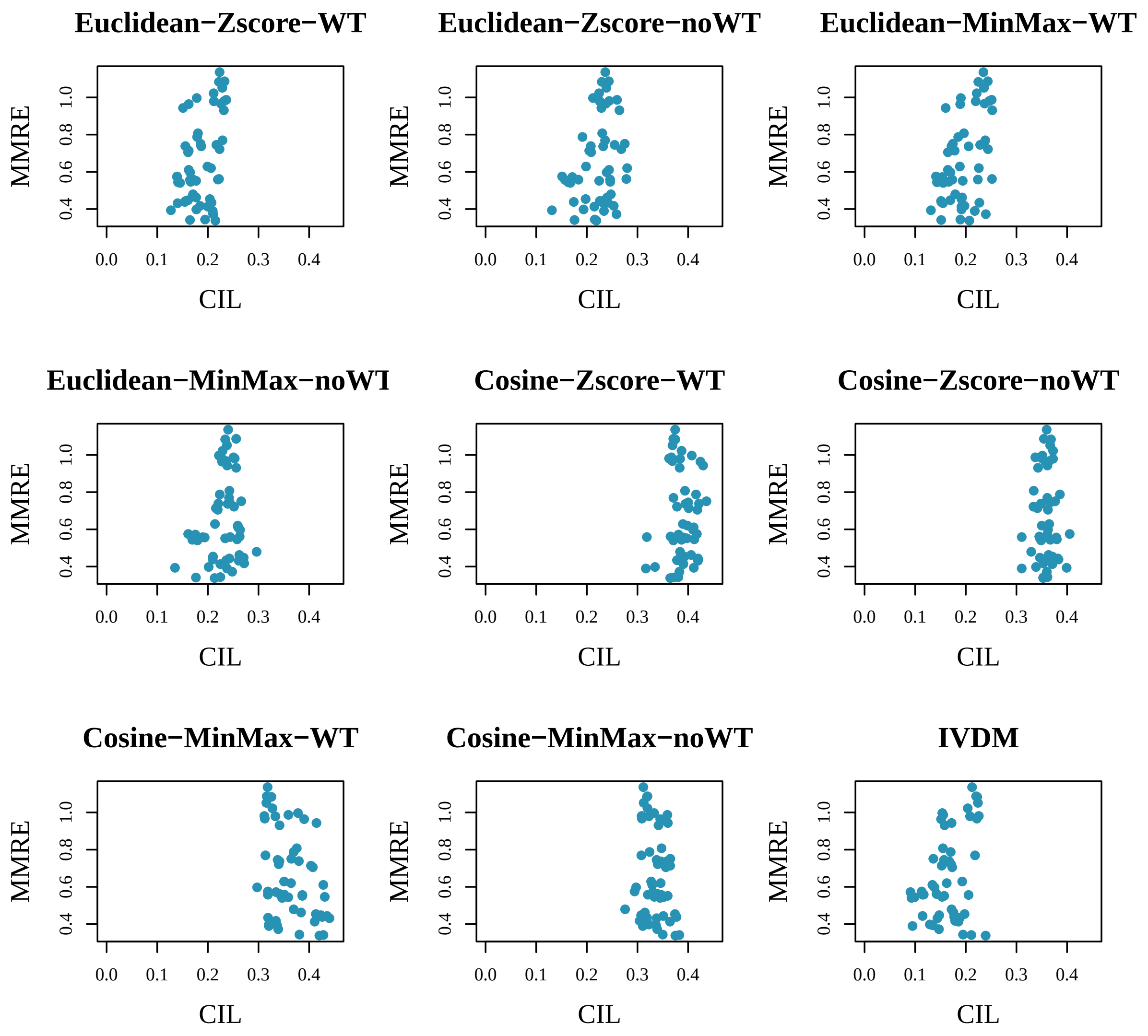


Figure A.8: The scatter plot of CIL vs MMRE for the estimation target of effort concerning Euclidean distance d_E, Cosine distance d_C and IVDM. Note that the threshold α is set to 0.3 for all plots.

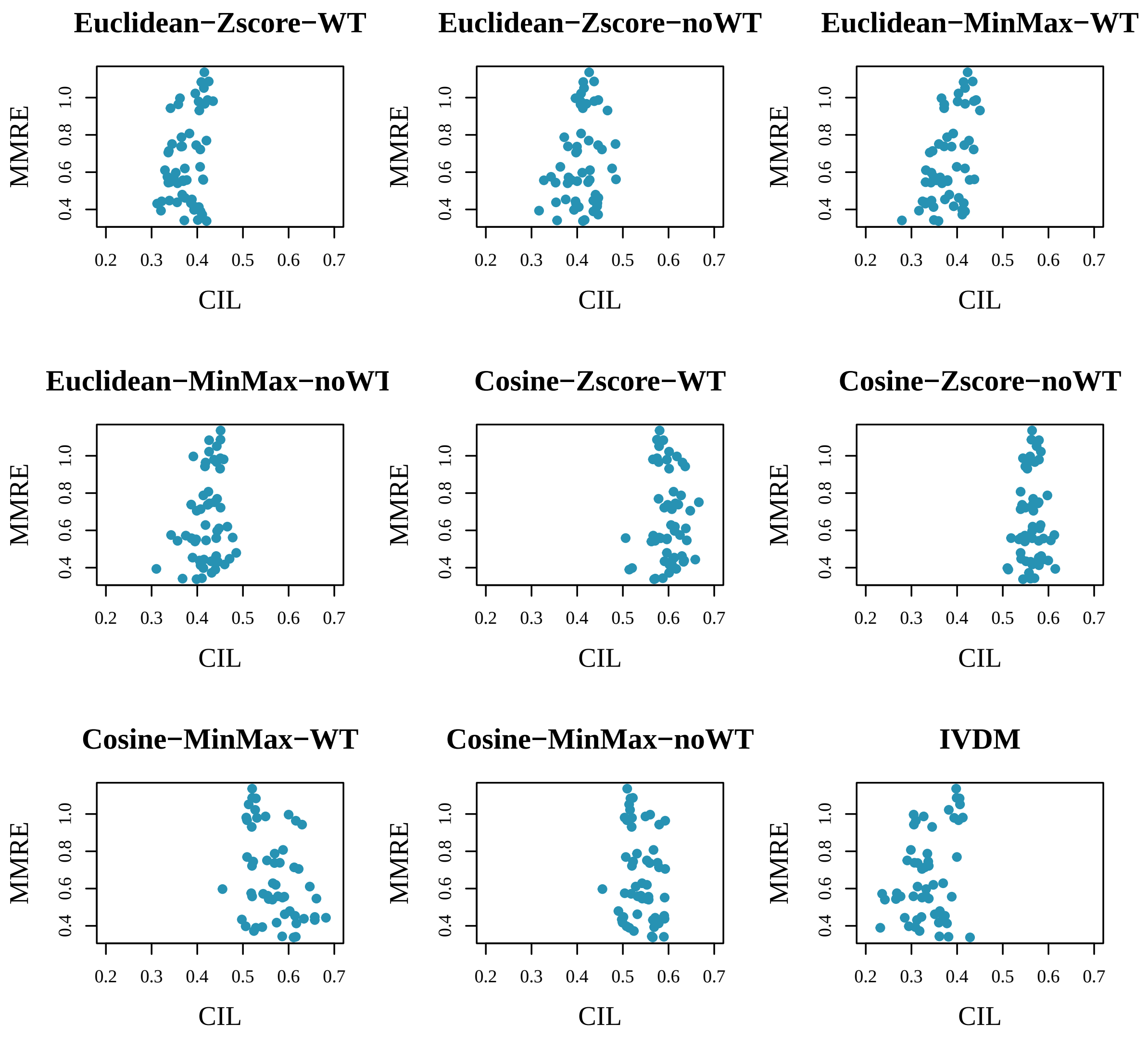


Figure A.9: The scatter plot of CIL vs MMRE for the estimation target of effort concerning Euclidean distance d_E, Cosine distance d_C and IVDM. Note that the threshold α is set to 0.5 for all plots.

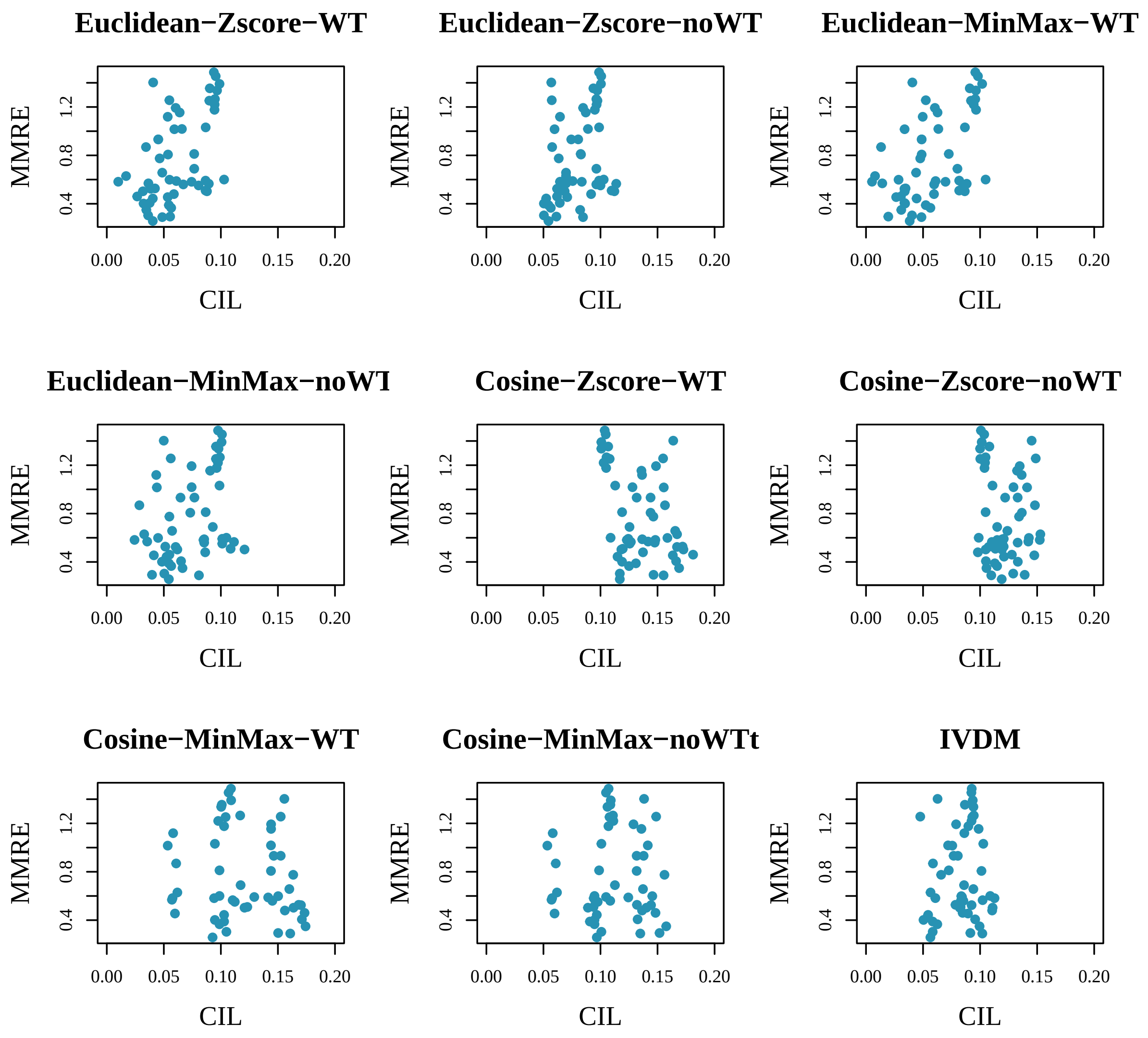


Figure A.10: The scatter plot of CIL vs MMRE for the estimation target of productivity concerning Euclidean distance d_E, Cosine distance d_C and IVDM. Note that the threshold α is set to 0.1 for all plots.

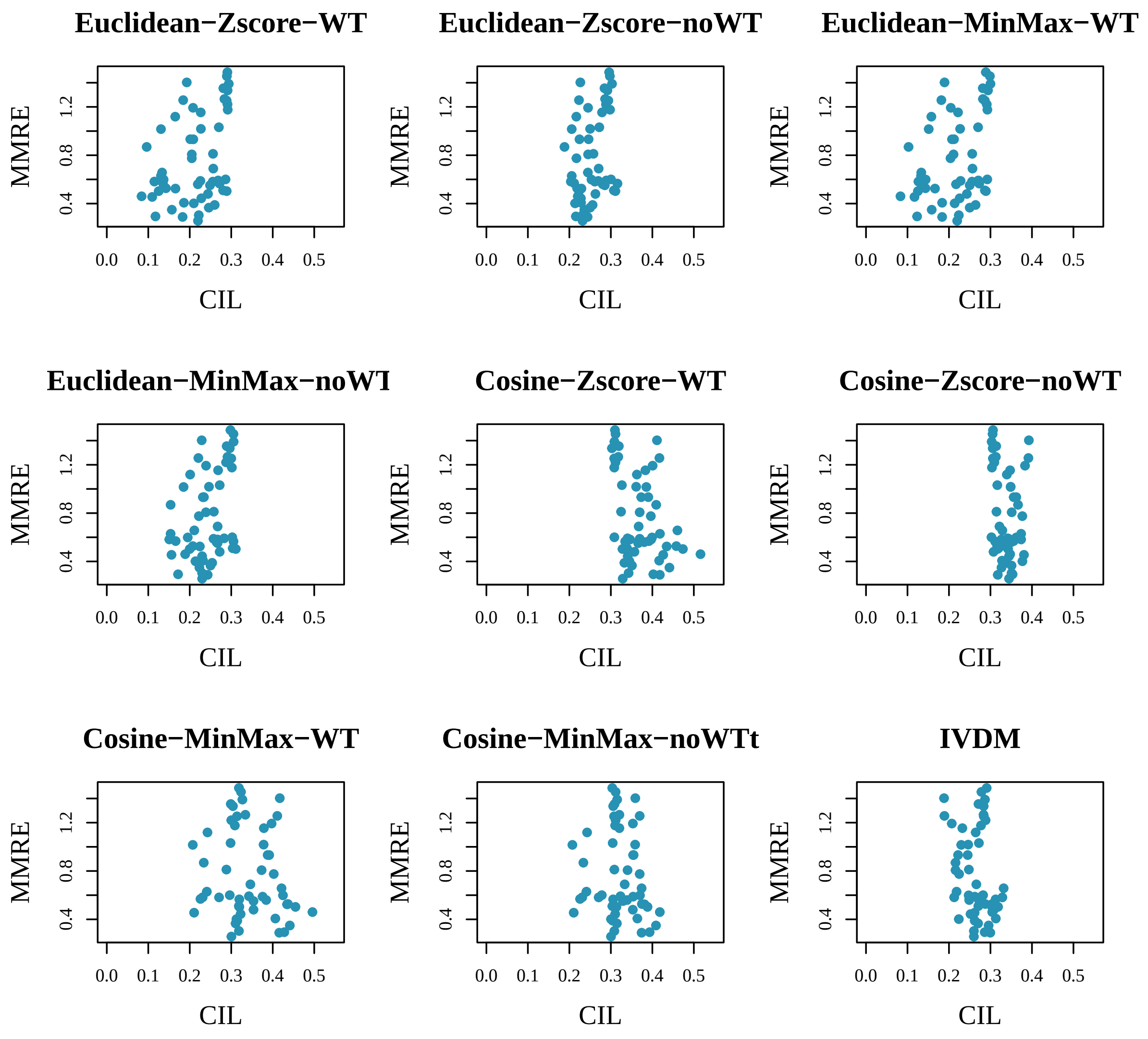


Figure A.11: The scatter plot of CIL vs MMRE for the estimation target of productivity concerning Euclidean distance d_E, Cosine distance d_C and IVDM. Note that the threshold α is set to 0.3 for all plots.

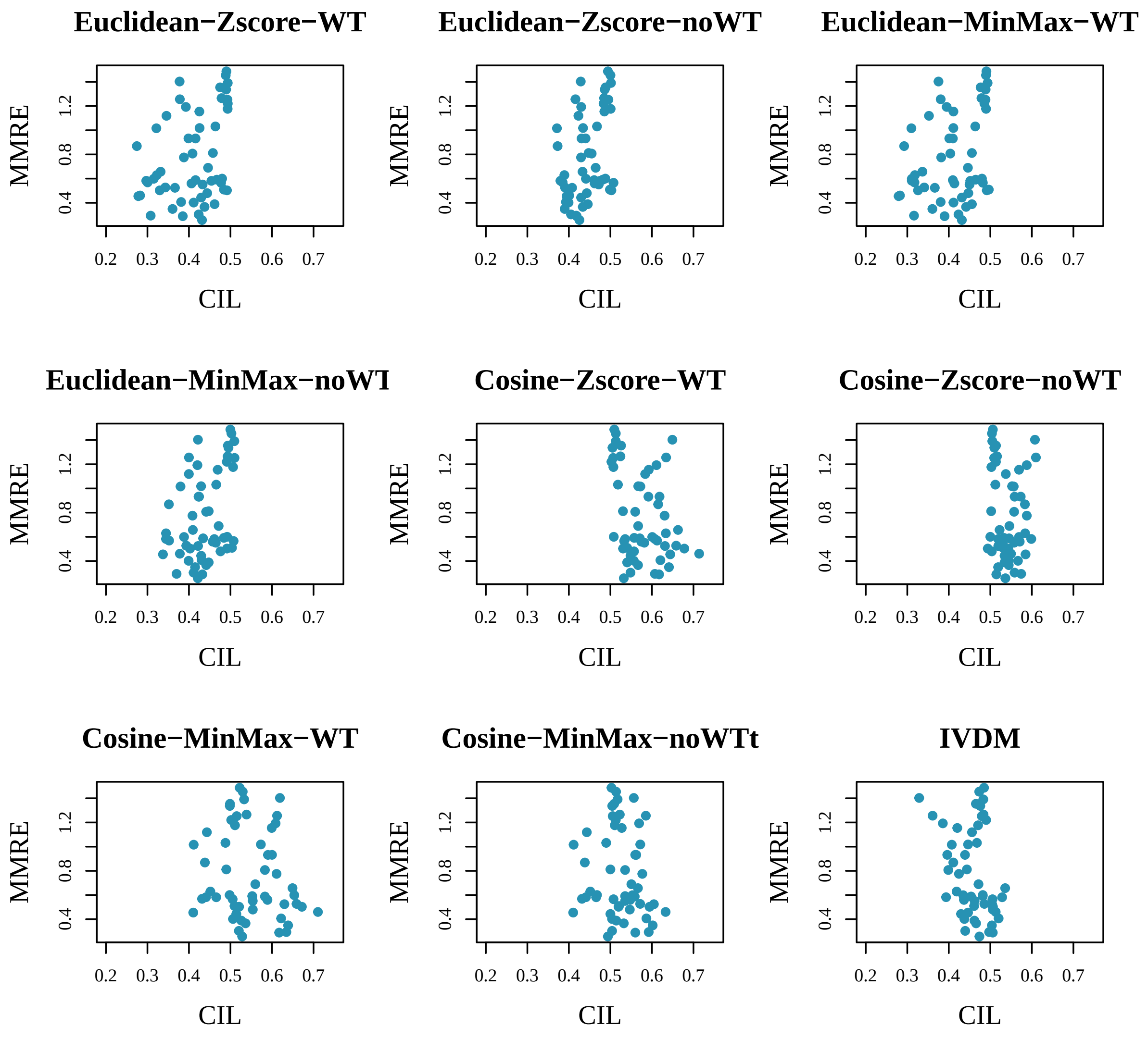


Figure A.12: The scatter plot of CIL vs MMRE for the estimation target of productivity concerning Euclidean distance d_E, Cosine distance d_C and IVDM. Note that the threshold α is set to 0.5 for all plots.

Appendix B

B.1 Toy example for computation of expected values of accuracy measures

Now let us study how this procedure works on a toy example. Consider the hypothetical test data with $T = 5$ modules shown in Table B.1. As shown in the first row of this table, A^{+} is 2 and A^{-} is 3.

The lower part of Table B.1 lists such predictions (i.e. vectors π_i) for which $P^{+} = 2$ and

Table B.1 An example of test data and possible predictions.

		1st module	2nd module	3rd module	4th module	5th module
Actual condition		1	1	0	0	0
Set of possible permutations Π	π_1	1	1	0	0	0
	π_2	1	0	1	0	0
	π_3	1	0	0	1	0
	π_4	1	0	0	0	1
	π_5	0	1	1	0	0
	π_6	0	1	0	1	0
	π_7	0	1	0	0	1
	π_8	0	0	1	1	0
	π_9	0	0	1	0	1
	π_{10}	0	0	0	1	1

$P^- = 3$[1]. As seen in this table, the number of prediction results with this composition is 10,

$$\#\Big(\Pi(2,3)\Big) = \binom{5}{2} = 10.$$

In addition, the expected value of π, i.e $\mathbf{E}(\Pi)$ can be computed as the average of all π_is as

$$\mathbf{E}(\Pi) = \frac{\sum\limits_{i=1}^{10} \pi_i}{10} = \Big[0.4 \quad 0.4 \quad 0.4 \quad 0.4 \quad 0.4\Big]. \tag{.1}$$

Without loss of generality, let us focus on the 1st module, which is an actual positive. There are 10 predictions for this module, i.e. one in each π_i, $i \in [1, 10]$. In addition, 2/5 of those 10 predictions are True Positive, i.e.

$$\frac{2}{5} \cdot 10 = 4.$$

Since there are 2 actual positives and there are 4 True Positives for each, the total number of True Positives concerning all the modules is

$$2 \cdot \frac{2}{5} \cdot 10 = 8.$$

Examining all predictions $\pi_1 \sim \pi_{10}$ in Table B.1, one may see that the total number of True Positives concerning all 5 modules is indeed (i.e. $TP = 8$). Therefore, the expected TP is computed as *TP per prediction*,

$$\mathbf{E}(TP) = \frac{8}{10}.$$

Similarly, examining $\pi_1 \sim \pi_{10}$, one can see that there are a total of 12 FPs, 18 TNs and 12 FNs in Table B.1. Thus, their expected values are $\mathbf{E}(FP) = 1.2$, $\mathbf{E}(TN) = 1.8$, $\mathbf{E}(FN) = 1.2$.

Based on these expected values, it is possible to compute the expected values of accuracy measures (e.g. $\mathbf{E}(\texttt{Precision})$, $\mathbf{E}(\texttt{Recall})$, $\mathbf{E}(\texttt{Npv})$, etc.). For example, concerning the example illustrated in Table B.1,

$$\begin{aligned}\mathbf{E}(\texttt{Precision}) &= \frac{\mathbf{E}(TP)}{\mathbf{E}(TP) + \mathbf{E}(FP)} \\ &= \frac{0.8}{0.8 + 1.2} \\ &= 0.4.\end{aligned}$$

[1]In this table, we sorted the vectors π_i in decreasing order, as if they are binary numbers. Nevertheless, the order of sorting is not important for our problem.

Relating to the same toy example, one can calculate the expected value of recall $\mathbf{E}(\texttt{Recall})$ as, once again, 0.4.

By using such expected values as baseline criteria, the quality of a particular prediction model can be evaluated. For instance, a model which yields as prediction result the 2nd permutation in Table B.1) attains `precision` of 0.5 and a `recall` of 0.5, both of which are larger than the corresponding expected values. In that respect, this prediction model can be judged to be a good one.

Acknowledgments

I would like to express my heartfelt gratitude to Professor Akito Monden of the Faculty of Natural Science and Technology, Okayama University, for his advice and guidance in conducting this research. I would also like to thank Associate Professor Zeynep Yücel of the Faculty of Natural Science and Technology, Okayama University so much, for her patience and guidance in my research. Also, I would like to thank Professors Norikazu Takahashi, Manabu Ohta and Toshihiro Yamauchi, Assistant Professors Mariko Sasakura and Kinari Nishiura, the Faculty of Natural Science and Technology, Okayama University, for their careful supervision. Finally, I would like to thank all the colleagues and friends in the Software Measurement and Analytics Laboratory, Okayama University.

References

[1] H. Akaike, “A new look at the statistical model identification,” *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–723, 1974.

[2] M. Azzeh, “A replicated assessment and comparison of adaptation techniques for analogy-based effort estimation,” *Empirical Software Engineering*, vol.17, no.1-2, pp.90–127, 2012.

[3] A. J. Albrecht and J. E. Gaffney, “Software function, source lines of code, and development effort prediction: a software science validation,” *IEEE Transactions on Software Engineering*, no.6, pp.639–648, 1983.

[4] Z. Abdelali, H. Mustapha, N. Abdelwahed, “Investigating the use of random forest in software effort estimation,” *Procedia Computer Science*, vol. 148, pp. 343–352, 2019.

[5] G. Abaei, W. Z. Tah, J. Z. W. Toh, and E. S. J. Hor, “Improving software fault prediction in imbalanced datasets using the under-sampling approach,” *Proc. International Conference on Software and Computer Applications*, 2022, pp. 41–47.

[6] B. W. Boehm, “Software engineering economics,” *IEEE Transactions on Software Engineering*, vol. SE-10, no. 1, pp. 4–21, 1984.

[7] A. P. Bradley, “The use of the area under the ROC curve in the evaluation of machine learning algorithms,” *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[8] A. Banimustafa, “Predicting software effort estimation using machine learning techniques,” *Proc. International Conference on Computer Science and Information Technology*, pp. 249–256, 2018.

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

[10] Y. Bao, N. Ishii and X. Du, "Combining multiple K-nearest neighbor classifiers using difference distance functions," *Proc. International Conference on Intelligent Data Engineering and Automated Learning*, pp. 634–641, 2014.

[11] M. L. Brown and J. Kros, "Data mining and the impact of missing data," *Industrial Management & Data Systems*, vol. 103, no. 8, pp. 611–621, Nov. 2003.

[12] K. E. Bennin, J. Keung, A. Monden, Y. Kamei, N. Ubayashi, "Investigating the effects of balanced training and testing data sets on effort-aware fault prediction models," *Proc. IEEE Computer Software and Applications Conference*, pp.154-163, 2016.

[13] G. E. P. Box and M. E. Muller, "A note on the generation of random normal deviates," *The Annals of Mathematical Statistics*, vol.29, no.2, pp.610-611, 1958.

[14] M. F. Bosu and S. G. MacDonell, "A taxonomy of data quality challenges in empirical software engineering," *Proc. Australian Conference on Software Engineering*, pp. 97–106, 2013.

[15] M. F. Bosu and S. G. MacDonell, "Experience: Quality benchmarking of datasets Used in software effort estimation," *Journal of Data and Information Quality*, vol. 11, no. 4, pp. 1–38, 2019.

[16] E. Blanzieri and F. Ricci, "Probability based metrics for nearest neighbor classification and case-based reasoning," *Proc. International Conference on Case-Based Reasoning*, pp. 14–28, 1999.

[17] J.S. Chen and C.H. Cheng, "Software diagnosis using fuzzified attribute base on modified MEPA," *Proc. International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Advances in Applied Artificial Intelligence*, pp. 1270–1279, 2006.

[18] H. M. Chung and P. Gray, "Special section: Data Mining," *Journal of Management Information Systems*, vol. 16, no. 1, pp. 11–17, 1999.

[19] D. Chicco and G. Jurman, “The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation,” *BMC Genomics*, vol. 21, no. 1, pp. 1–13, 2020.

[20] J. M. Desharnais, “Analyse statistique de la productivitie des projects informatique a partie de la technique des point des function,” *Masters Thesis University of Montreal*, 1989.

[21] T. Fawcett, “ROC graphs: Notes and practical considerations for researchers,” *Machine Learning*, vol. 31, pp. 1–38, 2004.

[22] J. Gorodkin, “Comparing two K-category assignments by a K-category correlation coefficient,” *Computational Biology and Chemistry*, vol. 28, no.5–6, pp. 367–374, 2004.

[23] 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,” *Proc. Annual Conference on Evaluation and Assessment in Software Engineering*, pp. 96–103, 2011.

[24] S. Gupta and A. Gupta, “Dealing with noise problem in machine learning data-sets: A systematic review,” *Procedia Computer Science*, vol. 161, pp. 466–474, 2019.

[25] L. Guo, Y. Ma, B. Cukic, and H. Singh, “Robust prediction of fault-proneness by random forests,” *Proc. International Symposium on Software Reliability Engineering*, pp. 417–428, 2004.

[26] V. García, R. A. Mollineda, and J. S. Sánchez, “Theoretical analysis of a performance measure for imbalanced data,” *Proc. International Conference on Pattern Recognition*, pp. 617–620, 2010.

[27] N. Gayatri, S. Nickolas, A. Reddy, and R. Chitra, “Performance analysis of data mining algorithms for software quality prediction,” *Proc. International Conference on Advances in Recent Technologies in Communication and Computing*, pp. 393–395, 2009.

[28] M. Gan, K. Sasaki, A. Monden, and Z. Yücel, “Generation of mimic software project data sets for software engineering research,” *Proc. International Workshop on Quantitative Approaches to Software Quality*, pp.38-43, 2018.

[29] S. Gupta, G. Sikka, and H.K. Verma, “Recent methods for software effort estimation by analogy,” *ACM SIGSOFT Software Engineering Notes*, vol. 36, no. 4, pp. 1–5, Aug. 2011.

[30] H. He and E. A. Garcia, “Learning from imbalanced data,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[31] J. V. Hulse, T. M. Khoshgoftaar, C. Seiffert, and L. Zhao, “Noise correction using bayesian multiple imputation,” *Proc. IEEE International Conference on Information Reuse and Integration*, pp. 478–483, 2006.

[32] J. Huang and C. X. Ling, “Using AUC and accuracy in evaluating learning algorithms,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 299–310, 2005.

[33] S. Huda, K. Liu, M. Abdelrazek, A. Ibrahim, S. Alyahya, H. Al-Dossari, and S. Ahmad, “An ensemble oversampling model for class imbalance problem in software defect prediction,” *IEEE access*, vol. 6, pp. 24 184–24 195, 2018.

[34] J. A. Hanley and B. J. McNeil, “The meaning and use of the area under a receiver operating characteristic (ROC) curve,” *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[35] A. Halevy, P. Norvig and F. Pereira, “The unreasonable effectiveness of data,” *IEEE Intelligent Systems*, vol. 24, no. 2, pp. 8–12, 2009.

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

[37] L. A. Jeni, J. F. Cohn, and F. D. L. Torre, “Facing imbalanced data recommendations for the use of performance metrics,” *Proc. Humaine Association Conference on Affective Computing and Intelligent Interaction*, pp. 245–251, 2013.

[38] M. Jureczko and L. Madeyski, “Towards identifying software project clusters with regard to defect prediction,” *Proc. International Conference on Predictive Models in Software Engineering*, pp. 1–10, 2010.

[39] M. Jureczko and D. Spinellis, “Using object-oriented design metrics to predict software defects,” *Models and Methods of System Dependability. Oficyna Wydawnicza Politechniki Wrocławskiej*, pp. 69–81, 2010.

[40] C. F. Kemerer, “An empirical validation of software cost estimation models,” *Communinations of the ACM*, vol.30, no.5, pp.416-429, May 1987.

[41] M. Kantardzic, “Data mining: Concepts, models, methods and algorithms,” *John Wiley and Sons*, 2002.

[42] T. M. Khoshgoftaar, K. Gao, and N. Seliya, “Attribute selection and imbalanced data: Problems in software defect prediction,” *Proc. International Conference on Tools with Artificial Intelligence*, pp. 137–144, 2010.

[43] T. M. Khoshgoftaar, K. Gao, A. Napolitano, and R. Wald, “A comparative study of iterative and non-iterative feature selection techniques for software defect prediction,” *Information Systems Frontiers*, vol. 16, no. 5, pp. 801–822, 2014.

[44] A. Kaur and I. Kaur, “Empirical evaluation of machine learning algorithms for fault prediction,” *Lecture Notes on Software Engineering*, vol. 2, no. 2, p. 176, 2014.

[45] J. Keung, E. Kocaguneli, and T. Menzies, “Finding conclusion stability for selecting the best effort predictor in software effort estimation,” *Automated Software Engineering*, vol. 20, no. 4, pp. 543–567, Dec. 2013.

[46] Y. Kamei and A. Monden, “Software engineering data repository for research and education [online],” http://analytics.jpn.org/SEdata/, 2016, [Accessed 2022-09-04].

[47] M. Kubat and S. Matwin, “Addressing the curse of imbalanced training sets: One-sided selection,” *Proc. International Conference on Machine Learning*, vol. 97, no. 1, p. 179, 1997.

[48] B. Kitchenham and E. Mendes, “Why comparative effort prediction studies may be invalid,” *Proc. International Conference on Predictor Models in Software Engineering*, no.4 pp.1-5, 2009.

[49] E. Kocaguneli, T. Menzies, and J. Keung, “On the value of ensemble effort estimation,” *IEEE Transactions on Software Engineering*, vol.38, no.6, pp.1403-1416, 2012.

[50] Y. Kamei, S. Matsumoto, A. Monden, K.-i. Matsumoto, B. Adams, and A. E. Hassan, “Revisiting common bug prediction findings using effort-aware models,” *Proc. International Conference on Software Maintenance*, pp. 1–10, 2010.

[51] T. M. Khoshgoftaar, X. Yuan, and E. B. Allen, "Balancing misclassification rates in classification-tree models of software quality," *Empirical Software Engineering*, vol. 5, no. 4, pp. 313–330, 2000.

[52] S. Kim, H. Zhang, R. Wu, and L. Gong, "Dealing with noise in defect prediction," *Proc. International Conference on Software Engineering*, pp. 481-490, 2011.

[53] R. J. Lewis, "An introduction to classification and regression tree (CART) analysis," *Proc. Annual Meeting of the Society for Academic Emergency Medicine*, pp. 1–14, 2000.

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

[55] Y. Liu, J. Cheng, C. Yan, X. Wu, and F. Chen, "Research on the Matthews correlation coefficients metrics of personalized recommendation algorithm evaluation," *International Journal of Hybrid Information Technology*, vol. 8, no. 1, pp. 163–172, 2015.

[56] S. Liu, Z, GUO, Y, Li, C, Wang, L, Chen, Z, Sun, Y, Zhou and B, Xu, "Inconsistent defect labels: Essence, causes, and influence," *IEEE Transactions on Software Engineering*, 2022.

[57] G. A. Liebchen and M. Shepperd, "Software productivity analysis of a large data set and issues of confidentiality and data quality," *Proc. IEEE International Software Metrics Symposium*, pp. 46–48, 2005.

[58] G. A. Liebchen and M. Shepperd, "Data sets and data quality in software engineering," *Proc. International Workshop on Predictor Models in Software Engineering*, pp. 39–44, 2008.

[59] B. Lauro and R. Traverso, "Data fitness For integration," *Mimeo*, 2018.

[60] G. A. Liebchen, B. Twala, M. Shepperd, and M. Cartwright, "Assessing the quality and cleaning of a software project dataset: An experience report," *Proc. International Conference on Evaluation and Assessment in Software Engineering*, pp. 122–128, 2006.

[61] R. Li and S. Wang, "An empirical study for software fault-proneness prediction with ensemble learning models on imbalanced data sets," *Journal of Software*, vol. 9, no. 3, pp. 697–704, 2014.

[62] C. Liu, D. Yang, X. Xia, M. Yan, and X. Zhang, “A two-phase transfer learning model for cross-project defect prediction,” *Information and Software Technology*, vol. 107, pp. 125–136, 2019.

[63] B. W. Matthews, “Comparison of the predicted and observed secondary structure of T4 phage lysozyme,” *Biochimica et Biophysica Acta (BBA)-Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975.

[64] K. D. Maxwell, *Applied statistics for software managers.* Prentice Hall, 2002.

[65] T. Menzies, A. Dekhtyar, J. Distefano, and J. Greenwald, “Problems with precision: A response to comments on data mining static code attributes to learn defect predictors,” *IEEE Transactions on Software Engineering*, vol. 33, no. 9, pp. 637–640, 2007.

[66] T. Menzies, J. Greenwald, and A. Frank, “Data mining static code attributes to learn defect predictors,” *IEEE Transactions on Software Engineering*, vol. 33, no. 1, pp. 2–13, 2006.

[67] J. C. Munson and T. M. Khoshgoftaar, “The detection of fault-prone programs,” *IEEE Transactions on Software Engineering*, vol. 18, no. 5, p. 423, 1992.

[68] R. Malhotra and S. Kamal, “An empirical study to investigate oversampling methods for improving software defect prediction using imbalanced data,” *Neurocomputing*, vol. 343, pp. 120–140, 2019.

[69] A. Monden, J. Keung, S. Morisaki, Y. Kamei, and K.-i. Matsumoto, “A heuristic rule reduction approach to software fault-proneness prediction,” *Proc. Asia-Pacific Software Engineering Conference*, vol. 1, pp. 838–847, 2012.

[70] T. Menzies, R. Krishna, and D. Pryor, “The seacraft repository of empirical software engineering data.”
https://zenodo.org/ communities/seacraft, 2017.

[71] S. Morasca and L. Lavazza, “On the assessment of software defect prediction models via ROC curves,” *Empirical Software Engineering*, vol. 25, no. 5, pp. 3977–4019, 2020.

[72] Y. Miyazaki, M. Terakado, K. Ozaki, and H. Nozaki, “Robust regression for developing software estimation models,” *Journal of Systems and Software*, vol.27, no.1, pp.3-16, 1994.

[73] F. Z. Mocnik, A. Zipf, and H. Fan, “Data quality and fitness for purpose,” *Proc. AGILE Conference on Geographic Information Science*, pp. 9–12, 2017.

[74] A. Nadi and H. Moradi, “Increasing the views and reducing the depth in random forest,” *Expert Systems with Applications*, vol. 138, p. 112801, 2019.

[75] K. Ono, M. Tsunoda, A. Monden, and K. Matsumoto, “Influence of outliers on estimation accuracy of software development effort,” *IEICE Transactions on Information and Systems*, vol. 104, no. 1, pp. 91–105, 2021.

[76] D. M. W. Powers, “Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation,” *arXiv preprint arXiv:2010.16061*, vol. 2, pp. 37–63, 01 2020.

[77] P. Phannachitta, J. Keung, A. Monden, and K. Matsumoto, “A stability assessment of solution adaptation techniques for analogy-based software effort estimation,” *Empirical Software Engineering*, vol.22, no.1, pp.474-504, 2017.

[78] P. Phannachitta, A. Monden, J. Keung, K. Matsumoto, “Case consistency: A necessary data quality property for software engineering data sets,” *Proc. International Conference on Evaluation and Assessment in Software Engineering*, Article no. 19, Apr. 2015.

[79] F. Peters and T. Menzies, “Privacy and utility for defect prediction: Experiments with morph,” *Proc. International Conference on Software Engineering*, pp.189-199, 2012.

[80] F. Peters, T. Menzies, L. Gong, and H. Zhang, “Balancing privacy and utility in cross-company defect prediction,” *IEEE Transactions on Software Engineering*, vol.39, no.8, pp.1054-1068, 2013.

[81] M. W. Reynolds, A. Bourke, and N. A. Dreyer, “Considerations when evaluating real-world data quality in the context of fitness for purpose,” *Pharmacoepidemiology and drug safety*, vol. 29, no. 10, pp. 1316–1318, 2020.

[82] I. Stamelos, L. Angelis, P. Dimou and E. Sakellaris, “On the use of Bayesian belief networks for the prediction of software productivity,” *Information and Software Technology*, vol. 45, no. 1, pp. 51–60, Jan. 2003.

[83] K. Strike, K. El Emam, and N. Madhavji, “Software cost estimation with incomplete data,” *IEEE Transactions on Software Engineering*, vol. 27, no. 10, pp. 890–908, Oct. 2001.

[84] M. Sokolova, N. Japkowicz, and S. Szpakowicz, “Beyond accuracy, F-score and ROC: A family of discriminant measures for performance evaluation,” *Proc. Australasian Joint Conference on Artificial Intelligence*, pp. 1015–1021, 2006.

[85] K. Shah, H. Patel, D. Sanghvi, and M. Shah, “A comparative analysis of logistic regression, random forest and KNN models for the text classification,” *Augmented Human Research*, vol. 5, no. 1, pp. 1–16, 2020.

[86] C. Sun, A. Shrivastava, S. Singh and A. Gupta, “Revisiting unreasonable effectiveness of data in deep learning era,” *Proc. IEEE International Conference on Computer Vision*, pp. 843–852, 2017.

[87] R. Taylor, “Interpretation of the correlation coefficient: A basic review,” *Journal of Diagnostic Medical Sonography*, vol. 6, no. 1, pp. 35–39, 1990.

[88] A. Tharwat, “Classification assessment methods,” *Applied Computing and Informatics*, 2020.

[89] D. Tiwari and M. Kumar, “Social media data mining techniques: A survey,” *Information and Communication Technology for Sustainable Development*, vol. 933, pp. 183–194, 2020.

[90] A. G. P. Varshini, K. A. Kumari, D. Janani, and S. Soundariya, “Comparative analysis of machine learning and deep learning algorithms for software effort estimation,” *Journal of Physics Conference Series*, vol. 1767, no. 012019, Feb. 2021.

[91] J. Wu and S. Gao, “Software productivity estimation by regression and Naive-Bayes classifier - an empirical research,” *Proc. International Conference on Promotion of Information Technology*, pp. 20–24, Aug. 2016.

[92] J. Wen, S. Li, Z. Lin, Y. Hu, C. Huang, “Systematic literature review of machine learning based software development effort estimation models,” *Information and Software Technology*, vol. 54, issue. 1, pp. 41–59, Jan. 2012.

[93] D. R. Wilson and T. R. Martinez, "Improved heterogeneous distance functions," *Journal of Artificial Intelligence Research*, vol. 6, no. 1, pp. 1–34, Jan. 1997.

[94] T. Watanabe, A. Monden, Y. Kamei, and S. Morisaki, "Identifying recurring association rules in software defect prediction," *Proc. International Conference on Computer and Information Science*, pp. 1–6, 2016.

[95] X. Xuan, D. Lo, X. Xia, and Y. Tian, "Evaluating defect prediction approaches using a massive set of metrics: An empirical study," *Proc. Annual ACM Symposium on Applied Computing*, pp. 1644–1647, 2015.

[96] Z. Xu, S. Pang, T. Zhang, X.-P. Luo, J. Liu, Y.-T. Tang, X. Yu, and L. Xue, "Cross project defect prediction via balanced distribution adaptation based transfer learning," *Journal of Computer Science and Technology*, vol. 34, no. 5, pp. 1039–1062, 2019.

[97] Z. Yücel, "Implications of log-transformation on data statistics." https://yucelzeynep.github.io/pub/2019_10_appdx_ieice_gan.pdf, 2019.

[98] S. J. Yen and Y. S. Lee, "Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset," *Intelligent Control and Automation*, pp. 731–740, 2006.

[99] Q. Zhu, "On the performance of Matthews correlation coefficient (MCC) for imbalanced dataset," *Pattern Recognition Letters*, vol. 136, pp. 71–80, 2020.

[100] H. Zhang and X. Zhang, "Comments on "data mining static code attributes to learn defect predictors"," *IEEE Transactions on Software Engineering*, vol. 33, no. 9, pp. 635–637, 2007.

[101] Software Reliability Enhancement Center, "White paper on software development data in 2018-2019," SEC Books, 2018.

List of publications

[1] Maohua Gan, Kentaro Sasaki, Akito Monden, and Zeynep Yücel, “Generation of mimic software project data sets for software engineering research,” *Proc. 6th International Workshop on Quantitative Approaches to Software Quality* (QuASoQ 2018), pp.38-43, Dec. 2018.

[2] Maohua Gan, Zeynep Yücel, Akito Monden, Kentaro Sasaki, “Empirical Evaluation of Mimic Software Project Data Sets for Software Effort Estimation,” *IEICE Transactions on Information and Systems*, vol. E103.D, No. 10, pp. 2094-2103, Oct. 2020.

[3] Maohua Gan, Zeynep Yücel, Akito Monden, “Improvement and evaluation of data consistency metric CIL for software engineering datasets,” *IEEE Access*, vol. 10, pp.70053-70067, 2022.

[4] Maohua Gan, Zeynep Yücel, Akito Monden, “Neg/pos-normalized Accuracy Measures for Software Defect Prediction,” *IEEE Access*, vol. 10, pp.134580-134591, 2022.