A Drift Propensity Detection Technique to Improve the Performance for Cross-Version Software Defect Prediction

Md Alamgir Kabir*, Jacky W. Keung*, Kwabena E. Bennin† and Miao Zhang*
*Department of Computer Science, City University of Hong Kong, Hong Kong
†Department of Software Engineering, Blekinge Institute of Technology, Sweden
{makabir4-c, miazhang9-c}@my.cityu.edu.hk, Jacky.Keung@cityu.edu.hk, kwabena.ebo.bennin@bth.se

***Abstract*—In cross-version defect prediction (CVDP), historical data is derived from the prior version of the same project to predict defects of the current version. Recent studies in CVDP focus on subset selection to deal with the changes of the data distributions. No prior study has focused on training data arriving in streaming fashion across the versions where the significant differences between versions make the prediction unreliable. We refer to this situation as Drift Propensity (DP). By identifying DP, necessary steps can be taken (e.g., updating or retraining the model) to improve the prediction performance. In this paper, we investigate the chronological defect datasets and identify DP in the datasets. The no-memory data management technique is employed to manage the data distributions and a DP detection technique is proposed. The idea behind the proposed DP detection technique is to monitor the algorithm's error-rate. The DP detector triggers DP, warning, and control flags to take necessary steps. The proposed technique is significantly superior in identifying the distribution differences (p-value $<$ 0.05). The DP's identified in the data distributions achieve large effect sizes (Hedges' $g \geq 0.80$) during the pair-wise comparisons. We observe that if the error-rate exponentially increases, it causes DP, resulting in prediction performance deterioration. We thus recommend researches and practitioners to address DP in the chronological datasets. Due to its potential effects in the datasets, the prediction models could be enhanced to get the best results in CVDP.**

***Keywords*-software defect prediction, cross-version defect prediction, drift propensity, streaming data, two-window-based data distributions**

I. Introduction

Software defect prediction (SDP) is nowadays pervasive in both software engineering (SE) academia and industry [1]. The SDP models classify the modules that are likely to be faulty [13]. Efficient identification of defective modules help to allocate the testing resources [2]. It is inevitable to identify the faulty modules for producing a quality software product [3, 15].

In the past decade, several prediction models including machine learning and statistical approaches have been proposed [4–8]. Traditional SDP models utilize the historical data to predict defects of the unlabeled data [9]. For within-project defect prediction (WPDP), prediction models are trained on labeled data and tested on the unlabeled data within the same project. In WPDP, cross-validation techniques are used to partition the training and test datasets within the same project's version, and moreover, is called inner-version defect prediction (IVDP) [12]. However, the development of software projects are becoming more sophisticated and the development process experiences frequent changes over time [10]. For the projects with multiple versions, the prediction model is more appropriate if the training data is obtained from the previous version and conducts the prediction of the upcoming version. This is called cross-version defect prediction (CVDP).

The software development environment is an evolving process where a mature project undergoes multiple changes by incorporating new requirements when the version is updated [10]. During this process, the updated version becomes more complicated due to refactoring and module deletion from the previous version. This causes distribution differences among the versions of the software project [11, 12]. The studies of [11, 12, 31] argued that the data distribution differences among the versions make the prediction results misleading. Dong et al. [13] affirmed that the prediction results of a well-trained prediction model could be inaccurate if the historical data differs over time. However, in the domain of SE prediction research, not many researchers have investigated it. In the CVDP setting, it is recommended by the studies of [11, 12, 31] to consider the previous version of the contemplated project as a training dataset for getting a better prediction since the prior version yields the related distribution information [14]. These studies noticed a certain degree of differences in the data distributions, which impacts the prediction performance.

Recent studies of Xu et al. [31–33] considered the issue of data distribution differences of the versions for CVDP. However, no studies considered that data comes in streaming fashion, taking all the versions into account in the context of CVDP. According to streaming data analytics, the learning process becomes more complicated due to the non-stationary environment leading to changes in data distributions [24]. The changes in data distribution are referred to as concept drift in the field of data stream mining [16]. Ekanayake et al. [17, 18] investigated the phenomenon of concept drift and conducted an empirical study to assess the reason why the prediction accuracy fluctuates in SDP. However, the studies of [17, 18] mainly focused on finding the location

978-1-7281-7303-0/20/$31.00 ©2020 IEEE
DOI 10.1109/COMPSAC48688.2020.0-154

of significant changes in data distribution and not comprehensive in the context of CVDP. In our prior work [19], we assessed the significant effect of concept drift and provided a systematic study of the impact of concept drift in SDP. To the best of our knowledge, this is the first attempt to consider the data that comes in streaming fashion. We refer to this situation as Drift Propensity (DP): significant changes in data distribution among the versions and if the DP is not addressed in the data distribution, it results in poor prediction performance. Identification of DP is essential to take necessary steps for improving prediction performance in the cross-version scenario. Since the differences of data across the versions have adverse influences on the prediction performance of CVDP.

In this work, we address the concern as DP and propose a DP detection technique that identify the DP's in the chronological defect datasets. By recognizing DP's in the datasets, practitioners could take necessary steps to enhanced the prediction results. We conduct an empirical study on 26 versions of six projects from the PROMISE repository to evaluate the effectiveness of our proposed DP detection method.

We highlight the main contributions as follows:

1) We propose a DP detection method potentially identified the DP's where the prediction accuracy varies in the versions during the learning process. We trigger three flags (i.e., DP, warning, and control flag) to take necessary steps during the DP detection process.
2) We provide analysis by performing experiments on 26 versions of six benchmark datasets. To our best knowledge, this is the first effort to identify DP by conducting experiments in the cross-versions defect datasets.
3) We employ fisher's based statistical test to identify the significant DP values during the detection process. Furthermore, we use the scheme of two-window-based data management technique to manage the versions and to create the scenario of cross-version.
4) The effect size computation for the pair-wise comparison, Hedges'g, ensures that the identified DP's are meaningful and practically significant. Based on the results, we observe that an exponential increase of the error-rate addresses the DP's in the chronological defect datasets.

We inspire the use of proposed technique to detect DP in the chronological defect datasets for CVDP. Further, the DP 's are critical to classification problem due to inequality in the decision boundary, thereby influences the classification performance. By identifying DP, the classification performance could be improved. The exponential increases of error-rate cause DP in the versions of software defect datasets and consequently exhibit poor prediction performance.

The rest of this paper is structured as follows. Section II describes the related work on CVDP. Section III presents the definition of DP. The two-window-based DP detection process is described in section IV. We introduce fisher's based statistical significant DP detection process in the section V. In section VI, we determine the configuration settings of our experiment. We outline the experimental results in section VII. Section VIII points the potential threats and section IX concludes this work by outlining future direction.

II. Related Work

CVDP has drawn high interest because of its applicability to the real development scenario where the previous version is used to predict defects in the current version of the same project, and distribution differences among the versions influence the prediction performance. However, not many researchers have considered it.

Yang and Wen [34] examined the ridge and lasso regression model for investigating the performance of CVDP. By employing 11 projects with 41 versions, they tried to find the strong correlations among the predictors and found the existence of multicollinearity in the considered datasets. Their work focused on ranking the defects. They avoided the task of feature selection and specified that it loses the information during the selection process. However, they didn't focus on the issue of data distributions differences, which affects prediction performance.

Shukla et al. [14] considered CVDP in their experiment and explained it as the problem of multi-objective optimization with two objectives: maximizing recall with minimizing the cost of misclassification and quality assurance. They studied 11 projects containing 41 versions and revealed that the multi-objective regression outperformed the single-objective regression. This study didn't focus on the distribution differences among the versions in CVDP, and it's applicability of building prediction models in cross-version settings.

Bennin et al. [20] performed cross-release performance evaluation of 25 software projects comprised of only two releases by considering Norm(P_{opt}) effort-aware performance measure where M5 and k* outperformed 9 other prediction models. They observed the strong correlation between the accuracies for intra-version and cross-version. However, the conducted 11 models were not statistically significant. This study also didn't consider the distribution differences among the releases that make the prediction accuracies outdated.

Lu et el. [21] considered the data distribution differences and performed feature selection task by using active learning method for CVDP and tried to alleviate distribution differences. However, the selected modules were unable to act as representative modules of the recent version as claimed by Huang et al. [23].

Xu et al. [33] developed a framework by combining principal component analysis and hybrid active learning

for selecting informative and representative modules from the current versions and tried to alleviate the distribution differences. By experimenting on ten projects comprising of 31 releases, they attempted to improve the prediction performance and further improved the work of Lu et al. [21]. In this work, the selected modules were added to previous versions, which required extra effort in labeling the modules.

Later, Xu et al. [32] proposed a subset selection technique based on dissimilarity for CVDP. They selected the representative modules from the previous version and combined them with current versions. In the work of [33], they chose the modules from the current version and added them to the prior release by using a hybrid active learning approach. Similar to the work in [32], they selected the modules from the previous version and added them to the current version. In both ways, they tried to alleviate the distribution differences by combining the selected modules to previous and current version accordingly. However, it requires extra effort to label the selected modules in the combining procedure.

The studies of [14, 21, 34] developed the prediction model for cross-version projects by using all the modules of the previous version without taking the issue of distribution differences into account. Besides, the studies in [31–33] used data selection methods to alleviate the distribution differences. For combining the selected modules to the considered versions for training the model, it requires extra effort during the adding process. None of these studies considered that the data across the versions come in streaming fashion, where data drifting within the versions can influence the prediction performance in cross-version settings.

III. Drift Propensity

This section provides a formal definition of drift propensity (DP), which may manifest in various ways in the data distributions.

DP describes unforeseeable variations in the data distributions by changing the statistical properties over time. The changes might occur for the hidden changes of variables. Formally, DP can be defined as follows: for a classification problem, given in a time period $(0, t)$, with a set of data samples $D_{(0,t)} = \{d_0, d_1, ..., d_t\}$ in which each instance d contains (X, y) where X is feature with the class label $(y \in Y)$. DP occurs at the time t, if the distribution of $D_{(0,t)}$ is significantly different from the $D_{(t+1),\infty}(X, y)$ as simply defined as $D_{(0,t)}(X, y) \neq D_{(t+1),\infty}(X, y)$. DP is critical for classification problem. Particularly, in cross-version defect prediction, the data distribution comes over time and trained data is derived from the previous version of the defect datasets where current version is considered as test data. DP creates such differences that may lead to inequality in the decision boundaries, thereby decreasing learning accuracy.

In the data distributions, DP can occur by changing the probability F of X and y at time t. Fig. 1 portraits how

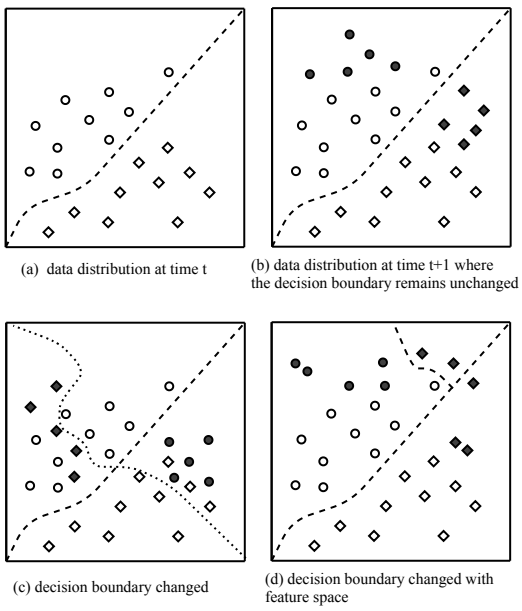

Figure 1. DP detection in target feature space by changing the decision boundary. White circle and diamond are the two class labels at time t where black circle and diamond of class labels at time $(t + 1)$.

DP occurs in a feature space. It may occur in the following ways (Fig. 1):

1) **Feature Space DP:** The decision boundary of Fig. 1(a) and Fig. 1(b) remain unchanged though $F_t(X) \neq F_{t+1}(X)$. In this case, $F_t(X)$ and $F_{t+1}(y|X)$ are same.
2) **Decision Boundary DP:** In Fig. 1(c), DP causes by changing the decision boundary where $F_t(X) \neq F_{t+1}(y|X)$. In this case, $F_t(X)$ and $F_{t+1}(X)$ are same.
3) **Feature Space with Decision Boundary DP:** In Fig. 1(d), DP occurs for the both cases, changing the feature space with decision boundary where $F_t(X) \neq F_{t+1}(X)$ and $F_t(X) \neq F_{t+1}(y|X)$.

IV. DP Detector

In this paper, we propose a DP detection technique to compare the two consecutive data distributions. As a base learner, we use naïve Bayes (NB) in our experimental study. For detecting DP in the chronological software defect datasets, we assume that the data comes in streaming fashion. In our proposed DP technique, we assume the two consecutive versions of a software project as two-window-based data distributions that arrive over time. In the CVDP, the prior version of a software project is considered as historical data, and the model is built on that data to predict defects for the next release of the software project. We employ the scheme proposed by drift detection method (DDM) [22] to identify DP in the chronological defect datasets in the CVDP settings. The idea behind the method is to monitor the

algorithm's error-rate. If the difference between two adjacent versions of defect datasets is significant, it indicates DP that exits in the examined versions. The remedy is to retrain or enhance the model again to get better performance.

In this method, we assume that the data comes over time. We observe the error-rate (r) from the incoming n data samples that contains the number of instances i. The data comes at time t with standard deviation $d_i = \sqrt{r_i(1-r_i)/i}$. The DP detector stores the values of error-rate r_i and standard deviation d_i during the learning process. When the new window arrives, the model is built and stored the error-rate with its standard deviation. The DP detector manages two registers for minimum values of error-rate r_{low} and standard deviation d_{low}. From the stored error-rate r_i and standard deviation d_i, the detector compares with r_{low} and d_{low} to the target version of datasets. This comparison helps to identify the data distribution differences of the the adjacent defect datasets. During the detection process, the DP detector triggers three flags to identify the DP in the data distributions: DP flag, warning flag, and control flag. As suggested by Gama et al. [22], the flag of DP is triggered, if $r_i + d_i \geq r_{low} + 3d_{low}$ and warning flag is triggered, if $r_i + d_i \geq r_{low} + 2d_{low}$. DP flag indicates that DP is detected, where the necessary actions are imminent to achieve more reliable prediction performance. By triggering the warning flag, it implies that the target distributions will tend to be drift. Otherwise, the distribution is stationary by triggering the control flag. If the DP exits on that version, the current model will be outdated, and it needs to be retrained or updated. Thereby, DP is detected by processing the data that arrives in bundles.

Algorithm 1 presents the pseudo-code of DP detector. The model is built on the old window O_t and tested on recent window R_t based data distributions. DP detector detects the differences in data distributions by treating the data as chunk by chunk. However, error-rate is calculated r_i with standard deviations d_i (lines 2 to 5) of the considered versions of the defect datasets. The minimum values of error-rate r_{low} and its standard deviation d_{low} are kept in register to compare it with the recent values. In this DP detection settings, more than two versions are considered for this learning process and also to make the scenario of cross-version of the software project. The DP detector triggers the flags to identify DP in the data distribution by executing lines 7 to 13.

Algorithm 1 Outline of *DP Detector*

Input:
The old window O_t and recent window R_t
Learning algorithm L
The sample of train and test i
Result:
DP F_{dp}, Warning F_w and Control F_c.
Process:

1: **for** $t = 1, 2, ...$ **do**
2: Build the learner on O_t
3: Learner predicts on R_t
4: Calculate error-rate r_i
5: Calculate $d_i = \sqrt{r_i(1-r_i)/i}$
6: Store r_i, d_i and find the minimum error-rate r_{low} and standard deviation d_{low} during the process
7: **if** $(r_i + d_i \geq r_{low} + 3d_{low})$ **then**
8: F_{dp} detected // DP flag
9: **else if** $(r_i + d_i \geq r_{low} + 2d_{low})$ **then**
10: F_w detected // warning flag
11: **else**
12: F_c // control flag
13: **end if**
14: **end for**

V. Fisher's based Statistical Significant DP Detection

Fisher's Exact test is proposed to analyze contingency table and considered as exact test [25]. This test doesn't lose precision because it is not dependent on approximation and is not influenced by data behavior (e.g., imbalanced data) [26]. This test can be applied to analyze $m \times n$ dimensional tables. It is generally used 2×2 contingency tables, which fit the case of two-window-based data distribution.

Assume, we have two distributions which indicate two windows: recent window and old window. The confusion matrix for the recent window and old window is given below (e.g., in Table I and II).

Table I
For recent window

	Positive Prediction	**Negative Prediction**
Actual Positive	q_1	r_1
Actual Negative	s_1	t_1

Table II
For old window

	Positive Prediction	**Negative Prediction**
Actual Positive	q_2	r_2
Actual Negative	s_2	t_2

The primary reason for adopting a statistical test is to ensure that DP is not caused by sampling error and the DP value is statistically significant. Fisher's Exact test is implemented for the two-window-based data distributions using the 2×2 contingency tables presented in Tables I and II and explained below.

Given labeled datasets X_i and X_j are transformed into

Table III
ADAPTED GENERALIZED CONTINGENCY TABLE FOR RECENT AND OLD WINDOW

Labels	Recent window	Old window	Total
# Errors	s	t	s + t
# Correct Classes	q	r	q + r
Total	q + s	r + t	q + r + s + t

vector space $V_i = (q_1, r_1, s_1, t_1)$ and $V_j = (q_2, r_2, s_2, t_2)$ of the recent and old window, accordingly in Table I and II. Most classification models are assessed employing a confusion matrix. Two-window-based data distributions (i.e., recent window and old window) where four types of outcomes are achieved as follows: q_1, q_2 are the faulty instances that are correctly classified as faulty; s_1, s_2 are non-faulty instances incorrectly classified as faulty; r_1, r_2 are the faulty instances incorrectly classified as not faulty; t_1, t_2 are non-faulty instances classified as not faulty. Therefore, for the recent window, the total number of correct classification $q = q_1 + t_1$ and misclassification $s = s_1 + r_1$. For the old window, the total number of correct classification $r = q_2 + t_2$ and misclassification $t = s_2 + r_2$.

In the Eq. 1 and 2, p denotes as probability matrix of any event, $m = 1 - p$, and ! is factorial of the data. From the Table III, the t occurrences in the independent events of $s + t$ can be expressed by $((s + t)!/(s! \times t!)) \times p^s \times m^t$. Similarly, the events of probability q in $q+r$ is determined by $((q+r)!/(q! \times r!)) \times p^q \times m^r$. According to 2×2 contingency table, the probability of the two windows can be presented by the Table III and determined by the Eq. 1. By using the combined analysis, Yates [27] and Fisher [26] proved that the Eq. 1 could be formed by the Eq. 2.

$$p = \frac{(q + r) \times (s + t)!}{q! \times r! \times s! \times t!} \times p^{q+s} \times m^{r+t} \tag{1}$$

$$p = \frac{\binom{q+r}{q} \times \binom{s+t}{s}}{\binom{n}{q+s}} = \frac{(q + r)! \times (s + t)! \times (q + s)! \times (r + t)!}{q! \times r! \times s! \times t! \times n!} \tag{2}$$

Assuming, the null hypothesis is that the distribution on both windows X_i and X_j is not similar if and only if the observed $p - values$ are less than the significant level. Furthermore, our concern is to check if the windows are affected by the DP or not. The null hypothesis that the error of both windows should be rejected if and only if the observed values of p (0.05) are less than at 5% significant level.

We calculate the $p - values$ from the Eq. 2.The factorial functions of each item from the data distributions $[0...n]$ where n is the seen examples. As suggested by Lu et al. [24], the size of the window can be determined by the user. For simplicity, without discarding any instance, we consider the full data of the two consecutive versions as old and new window of the data distributions. Thus, we implement the Fisher's Exact test for two-window-based data distributions to identify the significant DP of the considered datasets.

VI. EXPERIMENTAL METHODOLOGY

In this section, we describe the experimental configuration in detail containing the chronological software defect datasets, and the illustration of the DP detection process with the data management technique.

A. Benchmark datasets

In this empirical study, we consider 26 versions of six benchmark datasets extracted from the PROMISE Repository which were provided by Jureczko and Madeyski [29, 30]. These datasets have been used in the studies of CVDP [31–34]. The statistics of the considered datsets are shown in Table IV where #Modules, #DefectiveM, and %DefectiveM denote the number of modules, the number of faulty modules, and the ratio of defect, respectively. This study requires datasets that include several versions of the same projects to prepare the scenario of CVDP. The datasets comprise modules where each module is characterized by the label of defective or non-defective with 20 static code software metrics (Table V).

Table IV
SUMMARY OF SELECTED DATASETS EXTRACTED FROM PROMISE DATA REPOSITORY

Project	Version	#Modules	#DefectiveM	%DefectiveM
ant	1.3	126	20	15.9%
	1.4	178	40	22.5%
	1.5	293	32	10.9%
	1.6	352	92	26.1%
	1.7	745	166	22.3%
camel	1.0	339	13	3.8%
	1.2	608	216	35.5%
	1.4	872	145	16.6%
	1.6	965	188	19.5%
jedit	3.2	272	90	33.1%
	4.0	306	75	24.5%
	4.1	312	79	25.3%
	4.2	367	48	13.1%
	4.3	492	11	2.2%
poi	1.5	237	141	59.5%
	2.0	314	37	11.8%
	2.5	385	248	64.4%
	3.0	442	281	63.6%
xalan	2.4	723	110	15.2%
	2.5	803	387	48.2%
	2.6	885	411	46.4%
	2.7	909	898	98.8%
xerces	init	162	77	47.5%
	1.2	440	71	16.1%
	1.3	453	69	15.2%
	1.4	588	437	74.3%

Table V
DESCRIPTION OF METRICS USED IN THE CHRONOLOGICAL DEFECT DATASETS

Abbreviation	Description
WMC	the sum of weighted methods in a class
DIT	the level of depth of inheritance tree for a class
NOC	the number of direct successor of a class
CBO	the number of coupling classes between objects
RFC	the number of methods responded from where received messages
LCOM	the set of methods not related to share the class fields
Ca	the number of other classes uses the specific class
Ce	the number of other classes are used by specific class
NPM	the number of declared public methods
LCOM3	a variant of LCOM
LOC	the number of line of code
DAM	the number of protected attributes declared
MOA	measuring the part-whole relationship
MFA	measuring the number of methods inherited meaning functional abstraction
CAM	cohesion among methods of class
IC	inheritance coupling
CBM	measuring the number of methods where methods are coupled
AMC	average method complexity
MaxCC	the maximum values of cyclomatic complexity of methods in a class
Avg(CC)	the average number of arithmetic mean of cyclomatic complexity of methods in a Class
BUG	bugs in the class

B. Evaluation setup

In this section, we describe the evaluation process of our empirical study. Learning from such environment where the data comes over time is very challenging task [28]. In data stream mining, the drift detection techniques are limited, in the sense that the methods are performed over every data instance in the streaming and are threshold dependent [35].

Table VI
CONFUSION MATRIX SETUP FOR TWO-CLASS PROBLEM

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

However, this study uses the DP detection process (see Section IV) to identify the DP in the chronological datasets. Following the DP detector method, error-rate is computed for each version by building the prediction model where train data is used as previous version and tested the classification model on the new version. Most of the models are evaluated from the confusion matrix and error-rate is computed described in Table VI where $error - rate = 1 - accuracy$. Accuracy is computed as $\frac{(TP+TN)}{(TP+TN+FP+FN)}$ where True Positives (TP) are the defective modules that are correctly classified, False Positives (FP) are the non-defective modules that are incorrectly classified, True Negatives (TN) are the non-defective modules as non-defective, and False Negatives (FN) are the defective modules that are incorrectly classified as non-defective, respectively.

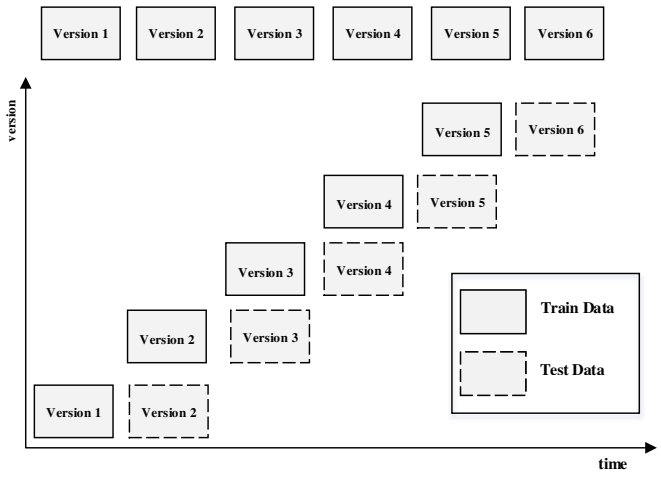

Figure 2. Adopted no-memory data management approach to manage the versions of the chronological defect datasets.

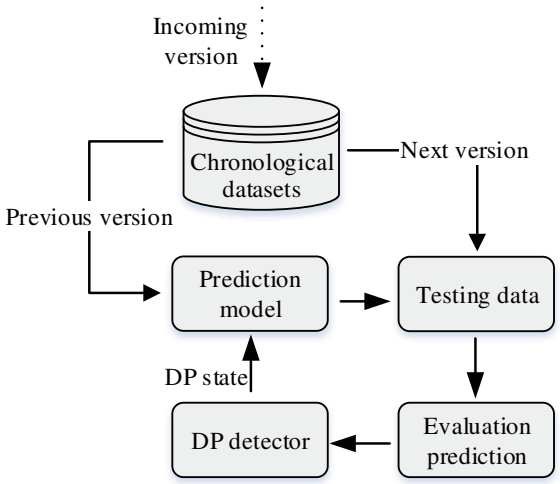

Figure 3. Evaluation process

Our proposed DP detection approach operates the data based on the two-window-based data management technique. We adopt the data processing scheme of no-memory data management technique to handle the data and execute it as two-window (see Fig. 2). In Fig. 2, versions of the data

distributions that shifts over time are observed by modifying the version update. Cross-version scenario is designed by considering the two adjacent versions as portrayed in Fig. 2. As an example, the model is trained by employing poi 1.5 and tested on poi 2.0. Our evaluation process adopts the single base learner naïve Bayes (NB) to compute the error-rate. The evaluation process is illustrated in Fig. 3. In the learning process, error-rate is monitored by triggering the flags as proposed in Algorithm 1. As mentioned above, the proposed DP detector triggers three flags values. By triggering the DP flag, it implies that DP is detected in the examined version, where the necessary steps (e.g., updating or retraining the model) are imminent to obtain more reliable prediction performance. By triggering the warning flag, it suggests that the target distributions will tend to be drift. The control flag determines the stability of the environment (i.e., the distributions do not change over time). By executing fisher's based statistical significant test (Section V), we ensure that the DP's are not caused by sampling error, but they are statistically significant as well.

C. Effect size computation for two-sample case

Effect size (ES) is considered as a quantitative measure of magnitude that has been applied in many studies of behavioural science [36]. Carl et al.[37] recommended the use ES in empirical studies to obtain practical significant, and meaningful results.

Moreover, the conclusion extracted from hypothesis testing might be imprecise because of the sufficient number of experimental subjects and the $p - values$ may also become inaccurate. Thus, weak inferences are not evaluated as part of the experimental settings [38]. Therefore, it is also recommended by [38–41] to consider effect sizes in the experimental study of SE along with significant statistical test for getting practical and meaningful results. Kampenes and Sjøberg [38] mentioned that 1% increase in defect detection would be highly practical significance for critical software if the defects belong to more critical defects.

As a robust ES computation standardized method, Hedges'g [42, 43] is considered to affirm the practical significance of our empirical study. This test reveals the significant differences between two consecutive data distributions and indicates meaningful and practical significance. Furthermore, it is an unbiased method and helps to find the identified differences that are significant for the small size of data distributions [37]. It is calculated by employing the means and the sample based standard deviations of the two data groups where the ES component is indicated as Hedges's g [41]. Suppose two data distributions are considered as two sequential versions of the chronological defect datasets. These two sequential data distributions are treated as control and experimental condition where the ES is defined by the Eq. 3.

$$Hedges'g = \frac{X_{ov} - X_{rv}}{S_{pooled}} \tag{3}$$

Eq. 3 implies the ratio of the two means (X_{ov} and X_{rv}) of the sequential defect datasets considered as old and recent version in the control and experimental condition. The difference between the two means is divided by the pooled standard deviation S_{pooled}. Since the versions of the defect datasets are dissimilar in size from each other, we calculate the pooled standard deviation by the Eq. 4.

$$S_{pooled} = \sqrt{\frac{(n_{ov} - 1) * s_{ov}^2 + (n_{rv} - 1) * s_{rv}^2}{n_{ov} + n_{rv} - 2}} \tag{4}$$

In Eq. 4, n_{ov}, s_{ov}, n_{rv}, and s_{rv} are the number of subjects, measurement of standard deviation of old version, the number of subjects, and measurement of standard deviation of recent version, respectively. We consider the absolute values of Hedges'g that are interpreted as follows: the values of Hedges'$g < 0.20$ indicate as the small effect, the values of about 0.50 interpret as moderate effect, and the values of about 0.80 explain as large effects using Cohen's benchmarks [44] and also adopted in the studies of [45, 46].

VII. Experimental results and discussion

In this section, we discuss the results of this experimental study. The results are presented based on the proposed DP detection technique and identify statistically significant values of DP in the chronological defect datasets using Hedges'g effect size and fisher's based statistical test at 5% significant level.

In this experimental study, we consider defect datasets which had at least four versions to establish a solid scenario of cross-version and to perform more generalizable results. The aim of this empirical study is to detect the distribution differences among the versions that are responsible for DP in the defect datasets. Mentioned above, the versions of the defect datasets are managed by the no-memory data management approach (Figure 2) and to establish the scenario of CVDP. The predictive error-rates are computed to identity the DP in the datasets based on the proposed DP detector. Please note that the experiment is conducted very carefully specifically in selecting the data distributions of the versions and group comparisons to eliminate any randomness in the experimental settings. In this case, the no-memory data management approach plays a vital role in our study.

The datasets contain several versions considered as chronological where the versions are created over time (Table IV). Experimental results are illustrated by adopting a two-window-based approach. In Table VII, we observe the error-rate attained using the two consecutive versions where IVersion, #InsOWin, and #InsRWin refer the initial version, number of instances of old version, and recent version, respectively. The values of DP obtained from the experiment

Table VII
PERFORMANCE OBSERVATION OF THE DP DETECTION TECHNIQUE IN THE CHRONOLOGICAL DEFECT DATASETS. TWO-WINDOW-BASED FISHER'S EXACT TEST RESULTS AND EFFECT SIZES OF THE PAIRWISE COMPARISON FOR TWO CONSECUTIVE DATA DISTRIBUTIONS. LARGE EFFECT SIZES POINT TO A PRACTICAL SIGNIFICANCE IN FAVOR OF DP DETECTION.

Project	OVersion	#InsOWin	RVersion	#InsRWin	DP value	p-value	Pooled effect size Hedges'g
ant	1.3	125	1.4	178	**4.91**	**2.62E-03***	**1.526****
	1.4	178	1.5	293	**15.10**	**8.69E-10***	**0.856****
	1.5	293	1.6	351	2.40	0.157	0.046
	1.6	351	1.7	745	1.02	0.093	0.092
camel	1.0	339	1.2	608	**9.72**	**6.29E-07***	**0.834****
	1.2	608	1.4	872	**5.04**	**6.16E-05***	**0.805****
	1.4	872	1.6	965	1.02	0.0864	0.085
jedit	3.2	272	4.0	306	**10.20**	**0.032***	**1.902****
	4.0	306	4.1	312	2.42	0.101	0.019
	4.1	312	4.2	367	2.19	0.059	0.317
	4.2	367	4.3	492	0.97	0.061	0.438
poi	1.5	237	2.0	314	1.02	0.285	0.178
	2.0	314	2.5	385	**15.85**	**1.53E-03***	**1.262****
	2.5	385	3.0	442	1.51	0.114	0.017
xalan	2.4	723	2.5	803	**4.07**	**1.77E-08***	**0.850****
	2.5	803	2.6	885	**11.07**	**1.14E-03***	**1.457****
	2.6	885	2.7	909	2.34	0.798	0.035
xerces	init	162	1.2	440	**4.45**	**0.004***	**0.868****
	1.2	602	1.3	453	1.00	0.055	0.025
	1.3	1055	1.4	588	**27.34**	**1.07E-05***	**1.456****

Old version (OVersion); Recent version (RVersion); number of instances in old window or version (#InsOWin); number of instances in recent window or version (#InsRWin); Drift Propensity value (DP value). "*" donates the values of 5% statistical significance p. "**" denotes the practical significance at $Hedges'g \geq 0.80$ calculated for large effect size and unequal data distributions.

are illustrated in the Table VII. To summarize the results and for easy understanding, we use "*" to denote the significant values of the DP's in the associated versions. Further, we use "**" to indicate the practical significance using pair-wise comparison with effect size test across the versions of the defect datasets. The effect size Hedges'g test is considered along with significant statistical analysis to ensure practical and meaningful results, and to avoid misleading $p-values$. Since the experiment includes fewer subjects in the settings, it is recommended by [41] to consider ES test as part of the SE empirical study. The boldface values represent the best results gained from our proposed algorithm and illustrate the identification of DP's in the associated versions of the project. The DP flags are triggered on those versions according to our detection process. Results from the DP detection process by pair-wise comparison among the two adjacent data distributions, we identify DP in the ant 1.3 and 1.4 versions among the seven versions, which achieve the best significant values ($p-value \ll 0.001$). These versions gain large effect size (i.e., ≥ 0.80). In this case, the most guided practice is to update or retrain the model if DP is identified in the data distributions to get more reliable prediction performance. In addition, the flags of DP are triggered in the versions of 1.0 and 1.2 for camel, 3.2 for jedit, 2.0 for poi, 2.4 and 2.5 for xalan, and init and 1.3 for xerces, which are statistically significant and achieve the more substantial effect in pair-wise comparisons.

The proposed DP detection technique identifies the number of DP's and their positions across the versions of the defect datasets. It is worth mentioning that the adaptation of fisher's exact test is computationally costly because of astute inability of factorial calculation, as also identified as limitation in [26]. Fisher's based statistical test for significant DP detection itself, a worthy contribution, can be utilized for similar types of problems. In addition to this, it can be applied in two-window-based data distributions in offline cases as well. Apart from this, the effect size computation for the pair-wise comparison, Hedges'g, ensures that the identified DP's are meaningful and practically significant.

One can consider how DP can hinder the prediction performance of CVDP. To explain, please consider the following scenario. The poi project contains four versions. If the model is built using the data of poi 2.0 as historical data to predict defects for consecutive releases, the prediction model won't be able to detect defects accurately. Based on the results of our empirical study, we observe that the model needs to be updated for predicting defects for the version of 2.5 because the difference of the data distributions is statistically significant, causing DP in that version and resulting poor performance of prediction model. Therefore, we advise addressing DP in the versions of the chronological defect datasets for defect prediction.

VIII. THREATS TO VALIDITY

In the empirical study, it is necessary to be aware of threats to validity of the obtained results [47]. Basili et al.

[48] argued that it is challenging to make a general conclusion from an experiment in SE because the experimental process requires a large number of relevant context variables.

We consider 26 cross-versions defect datasets from six benchmarks projects. To make the cross-versions scenario, we take the datasets that have at least four versions. For managing the data, we consider a two-window-based data management technique, no-memory approach. Others could have different preferences to make the scenario and for managing the defect datasets of the versions.

In our empirical study, we consider a single base learner to execute our proposed DP detection method. One could explore other classifiers to know the prediction performances. The aim of this study is to recognize the significant distribution differences that make the prediction misleading. Another possible threat is the adopted statistical test. We calculate the magnitudes of differences using Hedges' g. One could consider the other post hoc tests to identify the DP's in the datasets. As a consequence, the results might be different, although our adopted techniques work better on such large datasets in CVDP.

IX. Conclusion

Cross-version defect prediction (CVDP) is a more realistic scenario for predicting defects of software projects compared with cross-project. Very few studies have been conducted in the domain of CVDP specifically data with different distributions. In this paper, we formalize the issue of data distributions difference of two consecutive versions of defect datasets as drift propensity (DP) in cross-version settings. In our work, we propose a DP detection method that identifies the distribution differences of two consecutive versions of the defect datasets. We conduct our empirical study on 26 versions of six projects. We observe that significant distribution differences affect the prediction performance. The considered statistical test ensure that the DP's are statistically significant and obtain large effect size by calculating the magnitudes of the data distribution differences. It is observed that the exponential increase of the error-rate indicates the DP's in the cross-version of chronological defect datasets. Therefore, the identification of DP's in the versions may be useful for building more reliable prediction techniques for CVDP.

In the future, we seek to find the sources of DP's in the defect datasets. We intend to investigate the proposed DP detection technique in other domains.

Acknowledgement

This work is supported in part by the General Research Fund of the Research Grants Council of Hong Kong (No.11208017) and the research funds of City University of Hong Kong (7005028, 7005217), and the Research Support Fund by Intel (9220097), and funding supports from other industry partners (9678149, 9440227, 9440180 and 9220103).

References

[1] K. Bennin, J. Keung, and A. Monden, "On the relative value of data resampling approaches for software defect prediction," Empirical Software Engineering, vol. 24, no. 2, pp. 602–636, 2019.

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

[3] S. Ghaffarian and H. Shahriari, "Software Vulnerability Analysis and Discovery Using Machine-Learning and Data-Mining Techniques: A Survey," ACM Computing Surveys (CSUR), vol. 50, no. 4, pp. 1–36, 2017.

[4] S. Hosseini, B. Turhan and D. Gunarathna, "A Systematic Literature Review and Meta-Analysis on Cross Project Defect Prediction," in IEEE Transactions on Software Engineering, vol. 45, no. 2, pp. 111-147, 1 Feb. 2019, doi: 10.1109/TSE.2017.2770124.

[5] F. Wu et al., "Cross-Project and Within-Project Semisupervised Software Defect Prediction: A Unified Approach," in IEEE Transactions on Reliability, vol. 67, no. 2, pp. 581-597, June 2018, doi: 10.1109/TR.2018.2804922.

[6] Q. Yu, J. Qian, S. Jiang, Z. Wu and G. Zhang, "An Empirical Study on the Effectiveness of Feature Selection for Cross-Project Defect Prediction," in IEEE Access, vol. 7, pp. 35710-35718, 2019, doi: 10.1109/ACCESS.2019.2895614.

[7] X. Xia, D. Lo, S. J. Pan, N. Nagappan and X. Wang, "HYDRA: Massively Compositional Model for Cross-Project Defect Prediction," in IEEE Transactions on Software Engineering, vol. 42, no. 10, pp. 977-998, 1 Oct. 2016, doi: 10.1109/TSE.2016.2543218.

[8] F. Zhang, Q. Zheng, Y. Zou and A. E. Hassan, "Cross-Project Defect Prediction Using a Connectivity-Based Unsupervised Classifier," 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE), Austin, TX, 2016, pp. 309-320, doi: 10.1145/2884781.2884839.

[9] S. Lessmann, B. Baesens, C. Mues and S. Pietsch, "Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings," in IEEE Transactions on Software Engineering, vol. 34, no. 4, pp. 485-496, July-Aug. 2008.

[10] B. Ghotra, S. McIntosh and A. E. Hassan, "Revisiting the Impact of Classification Techniques on the Performance of Defect Prediction Models," 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, Florence, 2015, pp. 789-800.

[11] Xu, Zhou et al., "Cross version defect prediction with representative data via sparse subset selection," in Proceedings of the 26th Conference on program comprehension, pp. 132–143, 2018.

[12] S. Amasaki, "Cross-Version Defect Prediction using Cross-Project Defect Prediction Approaches: Does it work?," in Proceedings of the 14th International Conference on predictive models and data analytics in software engineering, 2018, pp. 32–41.

[13] F. Dong, J. Lu, K. Li and G. Zhang, "Concept drift region identification via competence-based discrepancy distribution estimation," 2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE), Nanjing, 2017, pp. 1-7.

[14] S. Shukla, T. Radhakrishnan, K. Muthukumaran, and L. Neti, "Multi-objective cross-version defect prediction," Soft Computing, vol. 22, no. 6, pp. 1959–1980, 2018.
[15] S. Hosseini, B. Turhan, and M. Mäntylä, "A benchmark study on the effectiveness of search-based data selection and feature selection for cross project defect prediction," Information and Software Technology, vol. 95, pp. 296–312, 2018.
[16] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," ACM Computing Surveys (CSUR), vol. 46, no. 4, pp. 1–37, 2014.
[17] J. Ekanayake, J. Tappolet, H. C. Gall and A. Bernstein, "Tracking concept drift of software projects using defect prediction quality," 2009 6th IEEE International Working Conference on Mining Software Repositories, Vancouver, BC, 2009, pp. 51-60.
[18] J. Ekanayake, J. Tappolet, H. C. Gall, and A. Bernstein, "Time variance and defect prediction in software projects: Towards an exploitation of periods of stability and change as well as a notion of concept drift in software projects," Empir. Softw. Eng., vol. 17, no. 45, pp. 348389, 2012.
[19] M. A. Kabir, J. W. Keung, K. E. Bennin and M. Zhang, "Assessing the Significant Impact of Concept Drift in Software Defect Prediction," 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Milwaukee, WI, USA, 2019, pp. 53-58, doi: 10.1109/COMPSAC.2019.00017.
[20] K. E. Bennin, K. Toda, Y. Kamei, J. Keung, A. Monden and N. Ubayashi, "Empirical Evaluation of Cross-Release Effort-Aware Defect Prediction Models," 2016 IEEE International Conference on Software Quality, Reliability and Security (QRS), Vienna, 2016, pp. 214-221.
[21] H. Lu, E. Kocaguneli and B. Cukic, "Defect Prediction between Software Versions with Active Learning and Dimensionality Reduction," 2014 IEEE 25th International Symposium on Software Reliability Engineering, Naples, 2014, pp. 312-322.
[22] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," Advances In Artificial Intelligence - Sbia 2004, vol. 3171, pp. 286–295, 2004.
[23] S. Huang, R. Jin and Z. Zhou, "Active Learning by Querying Informative and Representative Examples," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 10, pp. 1936-1949, Oct. 2014.
[24] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under Concept Drift: A Review," IEEE Transactions on Knowledge and Data Engineering, vol. 31, no. 12, pp. 2346–2363, 2019.
[25] R. Fisher, "Statistical Methods for Research Workers, Biological Monographs and Manuals," Oliver and Boyd, London, England, 1934.
[26] D. R. de L. Cabral and R. S. M. de Barros, "Concept drift detection based on Fisher's Exact test," Information Sciences, vol. 442–443, pp. 220–234, 2018.
[27] F. Yates, "Contingency Tables Involving Small Numbers and the x^2 Test," Supplement to the Journal of the Royal Statistical Society, vol. 1, no. 2, pp. 217–235, 1934.
[28] K. Nishida and K. Yamauchi, "Detecting Concept Drift Using Statistical Testing," in Discovery Science: 10th International Conference, DS 2007 Sendai, Japan, October 1-4, 2007. Proceedings, vol. 4755, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 264–269.
[29] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," in Proceedings of the 6th International Conference on predictive models in software engineering, 2010, pp. 1–10.
[30] L. Madeyski and M. Jureczko, "Which process metrics can significantly improve defect prediction models? An empirical study," Software Quality Journal, vol. 23, no. 3, pp. 393–422, 2015.
[31] Xu, Zhou et al., "TSTSS: A two-stage training subset selection framework for cross version defect prediction," The Journal of Systems & Software, vol. 154, pp. 59–78, 2019.
[32] Xu, Zhou et al., "Cross version defect prediction with representative data via sparse subset selection," in Proceedings of the 26th Conference on program comprehension, 2018, pp. 132–143.
[33] Z. Xu, J. Liu, X. Luo and T. Zhang, "Cross-version defect prediction via hybrid active learning with kernel principal component analysis," 2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER), Campobasso, 2018, pp. 209-220.
[34] X. Yang and W. Wen, "Ridge and Lasso Regression Models for Cross-Version Defect Prediction," in IEEE Transactions on Reliability, vol. 67, no. 3, pp. 885-896, Sept. 2018.
[35] J. Vinagre, A. M. Jorge, C. Rocha and J. Gama, "Statistically robust evaluation of stream-based recommender systems," in IEEE Transactions on Knowledge and Data Engineering.
[36] L. Wilkinson, "Statistical Methods in Psychology Journals," American Psychologist, vol. 54, no. 8, pp. 594–604, 1999.
[37] E. Carl et al., "Psychological and pharmacological treatments for generalized anxiety disorder (GAD): a meta-analysis of randomized controlled trials," Cognitive Behaviour Therapy, vol. 49, no. 1, pp. 1–21, 2020 [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/16506073.2018.1560358
[38] V. Kampenes, T. Dybå, J. E. Hannay, and D. I. . Sjøberg, "A systematic review of effect size in software engineering experiments," Information and Software Technology, vol. 49, no. 11, pp. 1073–1086, 2007.
[39] T. Dybå, V. Kampenes, and D. I. . Sjøberg, "A systematic review of statistical power in software engineering experiments," Information and Software Technology, vol. 48, no. 8, pp. 745–755, 2006.
[40] B. A. Kitchenham et al., "Preliminary guidelines for empirical research in software engineering," in IEEE Transactions on Software Engineering, vol. 28, no. 8, pp. 721-734, Aug. 2002. doi: 10.1109/TSE.2002.1027796
[41] J. Miller, "Applying meta-analytical procedures to software engineering experiments," Journal Of Systems And Software, vol. 54, no. 1, pp. 29–39, 2000.
[42] L. V. Hedges and I. Olkin, Statistical methods for meta-analysis. Orlando: Academic Press, 1985.
[43] A. Vargha and H. D. Delaney, "A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and Wong," Journal of Educational and Behavioral Statistics, vol. 25, no. 2, pp. 101–132, 2000.
[44] J. Cohen, Statistical power analysis for the behavioral sciences, 2nd ed. Hillsdale, N.J.: L. Erlbaum Associates, 1988.
[45] F. Garramone et al., "Personality profile and depression in migraine: a meta-analysis," Neurological sciences, 2019.
[46] S.B. Goldberg et al., "The experimental effects of psilocybin on symptoms of anxiety and depression: A meta-analysis," Psychiatry Research, vol. 284, pp. 112749, 2020.
[47] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, Experimentation in Software Engineering, 2012th ed., vol. 9783642290442. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–236.
[48] V. R. Basili, F. Shull and F. Lanubile, "Building knowledge through families of experiments," in IEEE Transactions on Software Engineering, vol. 25, no. 4, pp. 456-473, July-Aug. 1999. doi: 10.1109/32.799939