Proceedings of the Second International Conference on Electronics and Sustainable Communication Systems (ICESC-2021)
IEEE Xplore Part Number: CFP21V66-ART; ISBN: 978-1-6654-2867-5

Empirical Validation of cross-version and 10-fold cross-validation for Defect Prediction

Ruchika Malhotra
Department of Software Engineering
Delhi Technological University
Delhi, India
ruchikamalhotra2004@yahoo.com

Shweta Meena
Department of Software Engineering
Delhi Technological University
Delhi, India
shweta82336@gmail.com

2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC) | 978-1-6654-2867-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICESC51422.2021.9533030

***Abstract*—Nowadays, the development of a prediction model is one of the most important research area. Prediction models are helpful in providing accurate results for unseen data or future data. The most important phase of the software development life cycle (SDLC) is testing. During testing, we face some uncertain behavior of the program due to the presence of defects. To remove these defects in the early stage of SDLC, we design Software Defect Prediction Model (SDPM). Although many SDPM have been developed using various Machine Learning (ML) and statistical technique, but the generalizability of results have prevailed because the developed models use the same dataset for training and testing. This study aims to develop SDPM using cross-version, which using two different versions of a project for training and testing. The historical labeled defect data of the previous version is used for the updated or upcoming version for defect prediction which is termed as Cross-Version Defect Prediction (CVDP). To complete experimentation, we have used 26 datasets from an open-source repository. The performance of SDPM is analyzed using performance metrics. The SDPM is also developed using 10-cross validation. In the end, the comparison of CVDP and 10-cross validation has been done using a statistical test. The aim of conducting this study is to analyze the applicability of cross-version defect prediction when a sufficient amount of data is not available for training and testing. According to statistical analysis, it has been observed that cross-version can be used if we have to test our prediction model for unseen projects or upcoming projects.**

Keywords—Defect Prediction, Machine Learning, Cross-Version, Cross-Validation, Cross-Project, Statistical Test

I. INTRODUCTION

The software project size is increasing everyday, it directly impacts the cost and complexity of software. Due to which the defects in the software are indispensable [1] [2]. The detection and correction of defects before the beta release of the software is not feasible to assure software quality of software. The defect prediction focuses on the identification of the defect-prone modules and files of a software project. Defect prediction helps researchers, practitioners, and testers to test suspicious modules on priority. The prediction models developed in existing studies are supervised classification models that are developed using labeled data of a project and can be used for prediction on unlabeled dataset of same project that is termed as Within-Project Defect Prediction (WPDP). There is a lack of data availability due to which an efficient and reliable defect prediction model could not be developed using WPDP. The model development using WPDP requires huge amount of training data to overcome this issue.

To resolve this limitation, an emerging concept of transfer learning is employed. In transfer learning [3], a prediction model is developed by utilizing the labeled data of other projects that is termed is Cross-Project Defect Prediction (CPDP). The performance of the model developed using the CPDP concept is not effective, due to data distribution differences across various projects. Existing studies in the literature created a prediction model by labeled data of earlier versions and predict the unlabeled module's data of upcoming or future versions of the same project that is termed as CVDP [4] [5]. In the upgraded version of any software some of the modules would be deleted, some of the modules would be added, some additional functionality would be added. But, in terms of data distribution difference, the differences across multiple versions of the same project are low in comparison to data distribution difference across cross-projects because the upcoming version carries a huge amount of information or data from the existing version of that software [6].

Consider Project X.1, Project X.2, and Project Y. If Project X.1 is used for training and testing then it comes under WPDP. If project X.1 is used for training and Project X.2 is used for testing then it comes under CVDP. If Project X is used for training and Project Y is used for testing, then it comes under CPDP. This study demonstrated the applicability of CVDP.

However, various studies exist in the literature for change and defect prediction using CVD [7] [8] [9], but no study discussed the applicability of CVDP for generalization. This paper aims at the investigation of CVDP for generalizability. In literature, existing studies developed defect prediction models using either cross-project or WPDP. But to enhance the generalizability of results and removing the bias this study analyzes the performance of cross-version and 10-fold cross-validation. Thus, for the sake of achieving our research goal, the formulated research questions are as follows:

RQ1: What is the predictive capability of defect prediction models developed using cross-version validation?

RQ2: What is the predictive capability of the defect prediction model developed using 10-fold cross-validation?

RQ3: What is the feasibility of implementing cross-version in comparison to the performance of the model developed using 10-fold cross-validation i.e. to what extent cross-version validation is feasible?

In this study, the SDPM is developed using various ML techniques. The seven ML techniques are used for developing prediction model such as Naïve Bayes (NB), Logistic Regression (LR), Multilayer Perceptron (MLP), AdaBoost (AB), Bagging, J48, and Random Forest (RF). The generalizability of the prediction model is analyzed using cross-version and 10-fold cross-validation. The organization of rest of the paper is as follows: Section II

discussed the related work. The independent variables, dependent variables used and the descriptive statistics of the dataset used in this study are summarized in Section III. In Section IV, the answers to research questions are discussed. Section V summarized the threats to the validity of this study. Section VI summarized the conclusion and future work.

II. RELATED WORK

To analyze the effect of WPDP, CPDP, HCPDP, and CVDP various studies have been conducted.

WPDP: In WPDP, the classifier model is developed for projects having a sufficient amount of labeled data for training and testing within the same project. The existing literature analyzed the performance of classifiers [10], [11] developed for WPDP, feature selection techniques [12], sampling methods [13], and model validation techniques [14].

CPDP: CPDP is mainly used for projects having lack of data for prediction or testing. The researcher proposed a transfer learning concept for the project with a lack of data. CPDP can be done using transfer learning by transferring knowledge from other projects i.e., source project for predicting other projects i.e., target project [15]. Instance transfer, function transfer, relational information transfer, and parameter transfer are types of transfer learning. Transfer Component Analysis (TCA) has been developed to balance the data distribution between source and target dataset [16]. A hybrid composition model has been developed by combining ensemble learning and genetic algorithm for CPDP [17].

HCPDP: In HCPDP, the feature sets of projects used for source and target datasets are different. Various studies exist for HCPDP with different approaches. Since the training and testing projects have a different feature set due to which data distribution is also different. The correlation analysis method has been demonstrated to construct mapping across source and target project data into a common feature space by focusing on similarity mapping [18]. Nam et al. conducted the study [19] in which the methods are employed to find out the features set having the most similarity and the features with the highest similarity are used for conducting HCPDP. According to the analysis of existing studies, WPDP performs better than CPDP and HCPDP. Due to data distribution between source and target dataset, HCPDP does not perform better than WPDP [20]–[23]. The HCPDP study conducted by Jing et al. [18] reported results only for 22 HCP pairs out of total of 90 pairs of source and target project and it was verified universally. The method proposed by Nam et al. [19] to find out the similarity matching pairs across source and target datasets failed in many cases due to low matching scores in comparison to the threshold.

CVDP: CVDP is used for projects having multiple versions. The experimentation has been done for CVDP, in which the prediction model utilizes the labeled modules of a prior version of the current project [24]. CVDP is a methodology between WPDP and HCPDP. Since software development is a dynamic process that results in changes of the data structure characteristics for each updated version, the upcoming or updated version utilizes a large amount of information from the existing version of the project. The data distribution difference across multiple versions of the same project is smaller than the data distribution across completely different projects. According to the above statement, CVDP is more beneficial for defect prediction across multiple versions of the same project.

III. INDEPENDENT AND DEPENDENT VARIABLES USED

All the variables used in this study are discussed in this section.

A. Independent Variables

Model development has two phases one is training and another one is testing. The dataset used for training a model comprises of independent/input/predictor (software metrics) and dependent/output/outcome variables (prediction of defect prone module/ prediction of change prone module). In this study, we have used 26 versions of 10 different software projects given by Madeyski and Jureczko [25]. Each project consists of a class file of JAVA projects. These modules consist of 20 software metrics and one binary predictor variable for the defect proneness of a module (if a module is defective, it is labeled as 1, if the module is non-defective it is labeled as 0). The detailed description of these modules is provided in an existing study [26]. Detailed statistics description of all the 26 versions of 10 projects is summarized in Table I. The independent variables in the dataset are Chidamber and Kemerer (C&K) object-oriented metrics.

Table I: Statistics of Dataset

Project	# of Modules	# of Defective Modules	% of Defective Modules
ant_ver1.6	352	92	26.10%
ant_ver1.7	745	166	22.30%
camel_ver1.0	329	13	3.95%
camel_ver1.2	608	216	35.50%
camel_ver1.4	872	145	16.60%
camel_ver1.6	965	188	19.50%
ivy_ver2.0	352	40	11.36%
jedit_ver3.2	272	90	33.08%
jedit_ver4.0	306	75	24.50%
jedit_ver4.1	312	79	25.30%
jedit_ver4.2	367	48	13.10%
jedit_ver4.3	492	11	2.20%
log4j_ver1.0	135	34	25.20%

log4j_ver1.1	109	37	33.90%
log4j_ver1.2	205	189	92.20%
lucene_ver2.0	195	91	46.70%
lucene_ver2.2	247	144	58.30%
lucene_ver2.4	340	203	59.70%
poi_ver2.5	385	248	64.42%
poi_ver3.0	442	281	63.60%
synapse_ver1.2	266	86	32.33%
xalan_ver2.4	723	110	15.20%
xalan_ver2.5	803	387	48.20%
xalan_ver2.6	885	411	46.40%
xalan_ver2.7	909	898	98.80%
xerces_ver1.4	573	426	74.35%

B. Independent and Dependent Variables

The independent variables are object-oriented metrics (OOM) in a used dataset. Any kind of fault and bug results in some uncertain situation or unexpected failure of software known as DEFECT. In this analysis, the dependent variable is a defect. This is a nominal variable that indicates the presence and absence of a defect in a module ("YES" or "1" represents defective modules and "NO" or "0" represents a non-defective module).

IV. RESULT ANALYSIS

The answers to the research questions formulated in Section I are addressed in this section to achieve the objective of this study.

A. RQ1: What is the predictive capability of defect prediction models developed using cross-version validation?

To answer this RQ, we have experimented using seven ML techniques on different versions of all 8 projects out of 10 projects, 2 projects have one version only. We have created a prediction model for all the 30 combinations and the AUC values have been noted down in Table II. From the AUC values, it has been observed that in some of the cases CVDP is outperformed. The results are statistically validated using the Friedman test followed by post-hoc analysis that is Wilcoxon signed-rank test. For the Friedman test, we have formed and tested the following hypothesis.

Null Hypothesis (H_o):- There is no significant difference among the performance of all ML techniques for CVDP.

Alternate Hypothesis (H_a): There is a significant difference among the performance of all ML techniques for CVDP.

The ranks obtained using Friedman test of all the ML techniques are shown in Table III. The significance value obtained using the Friedman test is 0.000 which is less than the significance value. Hence, H_o is rejected and alternate H_a is accepted.

Table II: AUC values for Cross-Version Defect Prediction

Techniques used source dataset → target dataset	NB	LR	MLP	AB	Bagging	J48	RF
ant_ver1.6 → ant_ver1.7	0.779	0.778	0.75	0.8	0.826	0.679	0.803
camel_ver1.0 → camel_ver1.2	0.49	0.586	0.56	0.589	0.558	0.544	0.574
camel_ver1.0 → camel_ver1.4	0.482	0.644	0.636	0.631	0.644	0.565	0.677
camel_ver1.0 → camel_ver1.6	0.552	0.63	0.618	0.639	0.638	0.552	0.651
camel_ver1.2 → camel_ver1.4	0.687	0.68	0.666	0.668	0.761	0.654	0.765
camel_ver1.2 → camel_ver1.6	0.642	0.589	0.593	0.619	0.656	0.621	0.677
camel_ver1.4 → camel_ver1.6	0.633	0.628	0.679	0.64	0.68	0.632	0.682
jedit_ver3.2 → jedit_ver4.0	0.751	0.776	0.788	0.755	0.8	0.756	0.824
jedit_ver3.2 → jedit_ver4.1	0.759	0.81	0.788	0.775	0.795	0.727	0.813
jedit_ver3.2 → jedit_ver4.2	0.785	0.805	0.787	0.804	0.825	0.738	0.833
jedit_ver3.2 → jedit_ver4.3	0.608	0.64	0.589	0.575	0.586	0.531	0.625
jedit_ver4.0 → jedit_ver4.1	0.799	0.811	0.763	0.808	0.839	0.667	0.836
jedit_ver4.0 → jedit_ver4.2	0.826	0.824	0.764	0.83	0.823	0.655	0.852
jedit_ver4.0 → jedit_ver4.3	0.628	0.702	0.672	0.645	0.631	0.611	0.617
jedit_ver4.1 → jedit_ver4.2	0.837	0.872	0.797	0.817	0.842	0.784	0.858
jedit_ver4.1 → jedit_ver4.3	0.629	0.669	0.643	0.586	0.599	0.606	0.688
jedit_ver4.2 → jedit_ver4.3	0.624	0.648	0.595	0.594	0.686	0.759	0.706
log4j_ver1.0 → log4j_ver1.1	0.84	0.802	0.802	0.803	0.783	0.748	0.785
log4j_ver1.0 → log4j_ver1.2	0.621	0.728	0.708	0.66	0.694	0.586	0.666

log4j_ver1.1 → log4j_ver1.2	0.635	0.626	0.593	0.64	0.504	0.413	0.516
lucene_ver2.0 → lucene_ver2.2	0.64	0.641	0.615	0.666	0.657	0.564	0.658
lucene_ver2.0 → lucene_ver2.4	0.68	0.704	0.65	0.671	0.702	0.636	0.699
lucene_ver2.2 → lucene_ver2.4	0.72	0.678	0.658	0.701	0.648	0.658	0.639
poi_ver2.5 → poi_ver3.0	0.766	0.677	0.597	0.613	0.614	0.548	0.619
xalan_ver2.4 → xalan_ver2.5	0.584	0.619	0.567	0.632	0.646	0.472	0.619
xalan_ver2.4 → xalan_ver2.6	0.756	0.702	0.586	0.707	0.693	0.46	0.69
xalan_ver2.4 → xalan_ver2.7	0.79	0.762	0.528	0.6	0.778	0.743	0.808
xalan_ver2.5 → xalan_ver2.6	0.581	0.635	0.71	0.688	0.725	0.672	0.688
xalan_ver2.5 → xalan_ver2.7	0.705	0.661	0.768	0.826	0.821	0.722	0.883
xalan_ver2.6 → xalan_ver2.7	0.835	0.841	0.778	0.9	0.903	0.868	0.908

Table III: Mean Rank of ML technique

ML Technique	Mean Rank
J48	1.93
MLP	3.23
NB	3.65
AB	4.22
LR	4.58
Bagging	4.82
RF	5.57

Wilcoxon signed-rank test is performed to validate the alternate hypothesis of the Friedman test. The Wilcoxon signed-rank test results are shown in Table IV. All the ML techniques are paired with other techniques, in such a manner, total of 21 pairs are formed. The S denotes a significant difference and NS denotes no significant difference among that pair. If the p-value is greater than the significance level (0.05), then it indicated that there is no significant difference among that pair. From Table IV, it has been observed that RF performed better than NB in CVDP. MLP and J48 performed better than LR. J48 and RF performed better than MLP. J48 performed better than Bagging and AB. Hence, RF performed better than J48 all other ML techniques for CVPD. Thus, RF can be used for CVDP in upcoming projects with multiple versions of the same project.

Therefore, for RQ1 we conclude that:
For CVDP, defect prediction models developed with the applications of J48, MLP, NB, AB, LR, bagging and RF gave AUC in the range of 0.49 to 0.908 overall 30 pairs. RF is outperformed in comparison to all other ML techniques used in a study for prediction defect-prone modules based on the Friedman test result.

Table IV: Wilcoxon Signed Rank test for CVDP

Technique	p-value	Difference (S/NS)
LR - NB	0.361	NS
MLP - NB	0.361	NS
ABABOOST - NB	0.361	NS
BAGGING - NB	0.045	NS
J48 - NB	0.009	NS
RANDOMFOREST - NB	0.002	S
MLP - LR	0.001	S
ABABOOST - LOG	1.000	NS
BAGGING - LOG	1.000	NS
J48 - LOG	0.002	S
RANDOMFOREST - LOG	0.265	NS
ABABOOST - MLP	0.100	NS
BAGGING - MLP	0.018	NS
J48 - MLP	0.000	S
RANDOMFOREST - MLP	0.002	S
BAGGING - ABABOOST	0.100	NS
J48 - ABABOOST	0.001	S
RANDOMFOREST - ABABOOST	0.026	NS
J48 - BAGGING	0.000	S
RANDOMFOREST - BAGGING	0.045	NS
RANDOMFOREST - J48	0.000	S

B. RQ2: What is the predictive capability of defect prediction model developed using 10-fold cross validation?

To answer this RQ, the performance of SDP is evaluated and analyzed on the same training and testing dataset. The performance of the evaluated model is noted down in Table V for all 26 datasets. NB does not perform better in comparison to other ML techniques. The AUC values obtained using NBL are low. Here, the prediction model is developed using 10-fold cross validation for all 26 datasets using 7 ML techniques. To validate the results are statistically validated using the Friedman test followed by post-hoc analysis that is Wilcoxon signed-rank test. The Friedman test is performed with a significance level of 0.05 and 6 degree of freedom. The hypothesis created and tested for the Friedman test is as follows:

Null Hypothesis (H_o):- For 10-fold cross validation, there is no significant difference in the efficiency of all ML techniques.

Alternate Hypothesis (H_a): For 10-fold cross validation, there is a significant difference in the efficiency of all ML techniques.

Table V: AUC values for 10-fold cross validation

Technique used source dataset → target dataset	NB	LR	MLP	AB	Bagging	J48	RF
ant_ver1.6 → ant_ver1.6	0.823	0.844	0.807	0.82	0.882	0.82	0.904
ant_ver1.7 → ant_ver1.7	0.814	0.823	0.8	0.843	0.88	0.738	0.912
camel_ver1.0 → camel_ver1.0	0.834	0.736	0.735	0.883	0.825	0.68	0.863
camel_ver1.2 → camel_ver1.2	0.59	0.654	0.666	0.649	0.805	0.714	0.853
camel_ver1.4 → camel_ver1.4	0.678	0.678	0.637	0.652	0.701	0.592	0.73
camel_ver1.6 → camel_ver1.6	0.69	0.72	0.735	0.724	0.855	0.762	0.885
ivy_ver2.0 → ivy_ver2.0	0.809	0.827	0.803	0.83	0.879	0.8	0.916
jedit_ver3.2 → jedit_ver3.2	0.781	0.855	0.864	0.851	0.879	0.803	0.925
jedit_ver4.0 → jedit_ver4.0	0.769	0.811	0.832	0.813	0.88	0.804	0.914
jedit_ver4.1 → jedit_ver4.1	0.795	0.848	0.829	0.79	0.87	0.794	0.905
jedit_ver4.2 → jedit_ver4.2	0.845	0.864	0.838	0.873	0.906	0.772	0.928
jedit_ver4.3 → jedit_ver4.3	0.707	0.75	0.819	0.896	0.855	0.719	0.92
log4j_ver1.0 → log4j_ver1.0	0.863	0.821	0.846	0.84	0.861	0.767	0.903
log4j_ver1.1 → log4j_ver1.1	0.832	0.877	0.872	0.904	0.882	0.788	0.914
log4j_ver1.2 → log4j_ver1.2	0.792	0.81	0.851	0.823	0.887	0.798	0.936
lucene_ver2.0 → lucene_ver2.0	0.75	0.765	0.743	0.771	0.826	0.712	0.863
lucene_ver2.2 → lucene_ver2.2	0.624	0.643	0.706	0.699	0.757	0.687	0.826
lucene_ver2.4 → lucene_ver2.4	0.756	0.792	0.81	0.77	0.85	0.694	0.893
poi_ver2.5 → poi_ver2.5	0.773	0.849	0.879	0.888	0.918	0.875	0.944
poi_ver3.0 → poi_ver3.0	0.8	0.824	0.818	0.883	0.906	0.831	0.932
synapse_ver1.2 → synapse_ver1.2	0.768	0.789	0.81	0.823	0.843	0.757	0.892
xalan_ver2.4 → xalan_ver2.4	0.757	0.801	0.812	0.815	0.881	0.766	0.916
xalan_ver2.5 → xalan_ver2.5	0.604	0.695	0.719	0.71	0.838	0.759	0.89
xalan_ver2.6 → xalan_ver2.6	0.781	0.805	0.821	0.828	0.9	0.814	0.924
xalan_ver2.7 → xalan_ver2.7	0.925	0.933	0.822	0.93	0.894	0.811	0.958
xerces_ver1.4 → xerces_ver1.4	0.858	0.931	0.943	0.954	0.967	0.901	0.974

The ranks obtained using Friedman test of all the ML techniques are shown in Table VI. The computed p-value obtained using the Friedman test is 0.000 which is less than the significance value. Hence, H_o is rejected and alternate H_a is accepted.

Wilcoxon signed-rank test is performed to validate the alternate hypothesis of the Friedman test. Table VI shows the Wilcoxon signed-rank test results. All the ML techniques are paired with other techniques, in such a manner, total 21 pairs are formed. The S denotes a significant difference and NS denotes no significant difference among that pair. If the p-value is greater than the significance level (0.05), then it indicated there is no significant difference among that pair.

Table VI: Mean Rank of ML technique

ML Technique	Mean Rank
NB	2.28
J48	2.28
LR	3.25
MLP	3.37
AB	4.23
BAGGING	5.70
RF	6.88

From Table VI, it has been observed that LR, AB, Bagging, RF performed better than NB. Bagging and RF performed better than LR. Bagging and RF performed better than MLP. Bagging, J48, and RF performed better than AB. J48 and RF performed better than Bagging. RF performed better than J48. Hence, RF outperformed all other ML techniques for 10-fold cross-validation. Therefore, for RQ2 we conclude that:

For 10-fold cross-validation, defect prediction models developed with the applications of J48, MLP, NB, AB, LR, bagging and RF gave AUC in the range of 0.59 to 0.974 overall 26 pairs. RF outperformed in comparison to all other ML techniques used in this paper for the prediction of defect-prone modules based on the Friedman test result.

Table VII: Wilcoxon Signed Rank test for 10-fold cross validation

Technique	p-value	Difference (S/NS)
LR - NB	0.000	S
MLP - NB	0.201	NS
AB - NB	0.001	S
Bagging - NB	0.000	S
J48 - NB	0.855	NS
RF - NB	0.000	S
MLP – LR	0.584	NS
AB – LR	0.018	NS
BAGGING – LR	0.000	S
J48 – LR	0.100	NS
RF – LR	0.000	S
AB - MLP	0.201	NS
BAGGING - MLP	0.000	S
J48 - MLP	0.006	NS
RF - MLP	0.000	S
Bagging – AB	0.001	S
J48 – AB	0.001	S
RF – AB	0.000	S
J48 - Bagging	0.000	S
RF - Bagging	0.000	S
RF - J48	0.000	S

C. RQ3: What is the feasibility of implementing cross-version in comparison to the performance of model developed using 10-fold cross validation i.e. to what extent cross version validation is feasible?

The performance of prediction models developed using various ML techniques using cross-version validation should not vary significantly from the model developed using ten-fold cross-validation by applying the same ML techniques to research the applicability of cross-version methodology to recognize defective modules of the software. Cross-version validation applicability to defect, prediction models, is limited due to the significant amount of difference. The following hypothesis is established and tested to prove it:

Null Hypothesis (H_o): The performance of CVDP and the defect prediction model produced using 10-fold cross validation are not significantly different.

Alternate Hypothesis (H_a): The performance of CVDP and the defect prediction model established using 10-fold cross-validation differs significantly.

To determine the feasibility of cross-version validation, the findings are statistically validated. On the results of seven ML techniques, we used a pairwise Wilcoxon test with a significance level of 0.05. The performance of seven ML techniques is analyzed individually for both cross-version and10-fold cross-validation with respective AUC values. The pair-wise Wilcoxon test results are presented in Table VIII for cross-version and10-fold cross-validation. Table VIII depicts the p-value obtained from the pair-wise Wilcoxon test of seven ML techniques for cross-version and 10-fold cross-validation. From Table VIII it has been observed that the p-value is less than that of significance level (0.05). Therefore, the null hypothesis is rejected and the alternate hypothesis is accepted. Thus, there exists a significant difference among the performance of cross-version and 10-fold cross-validation. As a result of the Wilcoxon test results for RQ3, it is concluded that:

The performance of defect prediction modules developed using cross-version and 10-fold cross-validation differ significantly. Thus, it has been observed that usage of existing versions of software for upcoming software is not feasible in comparison to 10-fold cross-validation for the prediction of defects in upcoming versions of software.

Table VIII: Paired Wilcoxon test on CVDP and model developed using 10 fold cross-validation

Technique	p-value
$NB^{CV} - NB^{TF}$	0.002
$LR^{CV} - LR^{TF}$	0.000
$MLP^{CV} - MLP^{TF}$	0.000
$AB^{CV} - AB^{TF}$	0.000
$Bagging^{CV} - Bagging^{TF}$	0.000
$J48^{CV} - J48^{TF}$	0.000
$RF^{CV} - RF^{TF}$	0.000

V. THREATS TO VALIDITY

The validity threats of this empirical study are discussed in this section. In this study, the open-source datasets of various software are used for experimentation. This may affect the generalizability of upcoming software in different languages. Conclusion threats are the threats that establish an incorrect relationship among the experimentation and its output. In this study, we have analyzed the results using a statistical test to increase the validation of results and it reduces the threat to conclusion validity. Construct validity refers to the degree to which the independent and dependent variables measure the principles that they are supposed to measure. C&K metrics are used as independent variables in this analysis, and they have successfully demonstrated their ability to predict software defects in similar literature. Thus, the threat to construct validity is reduced in this study with respect to the independent variables.

VI. CONCLUSION AND FUTURE WORK

In this study, the feasibility of cross-version is analyzed using 26 versions of 10 different projects. Two experiments are conducted in this study. In the first

experiment, the defect prediction model is developed using the cross-version validation method. The defect prediction model is developed using 10-fold cross-validation in the second experiment, and the performance of this model is compared to that of a model developed using cross-version. The results are validated using statistical test. The results obtained from paired Wilcoxon test proved that it is not feasible to implement the CVDP model in comparison to the model developed using 10-fold cross-validation. Cross-version helps in improving the quality of software by using the existing version of software to identify defects in the updated version of that project. It has been observed that the training and testing dataset of the same project is efficient instead of using different versions of a project as a training and testing dataset.

We will try to explore this study further for large projects developed in various programming languages. In future, we will replicate this study by establishing similar data distribution among training and testing projects. To improve the viability of the CVDP model, we intend to investigate techniques such as heuristic-based and hybridized techniques. We will also analyze the feasibility of CVDP in comparison to CPDP.

REFERENCES

[1] N. E. Fenton and N. Ohlsson, "Quantitative analysis of faults and failures in a complex software system," *IEEE Transactions on Software Engineering*, vol. 26, no. 8, pp. 797–814, Aug. 2000.

[2] J. C. Knight, "Safety critical systems: Challenges and directions," in *Proceedings - International Conference on Software Engineering*, 2002, pp. 547–550.

[3] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10. pp. 1345–1359, 2010.

[4] E. Arisholm and L. C. Briand, "Predicting fault-prone components in a Java legacy system," in *ISESE'06 - Proceedings of the 5th ACM-IEEE International Symposium on Empirical Software Engineering*, 2006, vol. 2006, pp. 8–17.

[5] P. He, B. Li, X. Liu, J. Chen, and Y. Ma, "An empirical study on software defect prediction with a simplified metric set," *Information and Software Technology*, vol. 59, pp. 170–190, Mar. 2015.

[6] H. K. Dam, T. Tran, T. Pham, S. W. Ng, J. Grundy, and A. Ghose, "Automatic Feature Learning for Predicting Vulnerable Software Components," *IEEE Transactions on Software Engineering*, vol. 47, no. 1, pp. 67–85, Jan. 2021.

[7] R. Malhotra and M. Khanna, "Inter Project Validation for Change Proneness Prediction using Object Oriented Metrics," *Software engineering : An International Journal*, vol. 3, no. 1, pp. 21–31, 2013.

[8] R. Malhotra and A. J. Bansal, "Cross project change prediction using open source projects," in *Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2014*, 2014, pp. 201–207.

[9] Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang, "An investigation on the feasibility of cross-project defect prediction," *Automated Software Engineering*, vol. 19, no. 2, pp. 167–199, Jun. 2012.

[10] B. Ghotra, S. McIntosh, and A. E. Hassan, "Revisiting the impact of classification techniques on the performance of defect prediction models," in *Proceedings - International Conference on Software Engineering*, 2015, vol. 1, pp. 789–800.

[11] 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*, 2008, vol. 34, no. 4, pp. 485–496.

[12] Z. Xu, J. Liu, Z. Yang, G. An, and X. Jia, "The Impact of Feature Selection on Defect Prediction Performance: An Empirical Comparison," in *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, 2016, pp. 309–320.

[13] K. E. Bennin, J. Keung, P. Phannachitta, A. Monden, and S. Mensah, "MAHAKIL: Diversity Based Oversampling Approach to Alleviate the Class Imbalance Issue in Software Defect Prediction," *IEEE Transactions on Software Engineering*, vol. 44, no. 6, pp. 534–550, Jun. 2018.

[14] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "An Empirical Comparison of Model Validation Techniques for Defect Prediction Models," *IEEE Transactions on Software Engineering*, vol. 43, no. 1, pp. 1–18, Jan. 2017.

[15] S. Herbold, A. Trautsch, and J. Grabowski, "A Comparative Study to Benchmark Cross-Project Defect Prediction Approaches," *IEEE Transactions on Software Engineering*, vol. 44, no. 9, pp. 811–833, Sep. 2018.

[16] J. Nam, S. J. Pan, and S. Kim, "Transfer defect learning," in *Proceedings - International Conference on Software Engineering*, 2013, pp. 382–391.

[17] X. Xia, D. Lo, S. J. Pan, N. Nagappan, and X. Wang, "HYDRA: Massively compositional model for cross-project defect prediction," *IEEE Transactions on Software Engineering*, vol. 42, no. 10, pp. 977–998, Oct. 2016.

[18] X. Jing, F. Wu, X. Dong, F. Qi, and B. Xu, "Heterogeneous cross-company defect prediction by unified metric representation and CCA-based transfer learning," in *2015 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings*, 2015, pp. 496–507.

[19] J. Nam, W. Fu, S. Kim, T. Menzies, and L. Tan, "Heterogeneous Defect Prediction," *IEEE Transactions on Software Engineering*, vol. 44, no. 9, pp. 874–896, Sep. 2018.

[20] F. Rahman, D. Posnett, and P. Devanbu, "Recalling the 'imprecision' of cross-project defect prediction," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, FSE 2012*, 2012, p. 1.

[21] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B.

Murphy, "Cross-project defect prediction: A large scale experiment on data vs. domain vs. process," in *ESEC-FSE'09 - Proceedings of the Joint 12th European Software Engineering Conference and 17th ACM SIGSOFT Symposium on the Foundations of Software Engineering*, 2009, pp. 91–100.

[22] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empirical Software Engineering*, vol. 14, no. 5, pp. 540–578, Oct. 2009.

[23] L. C. Briand, W. L. Melo, and J. Wüst, "Assessing the applicability of fault-proneness models across object-oriented software projects," *IEEE Transactions on Software Engineering*, vol. 28, no. 7, pp. 706–720, Jul. 2002.

[24] H. Lu, E. Kocaguneli, and B. Cukic, "Defect prediction between software versions with active learning and dimensionality reduction," in *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, 2014, pp. 312–322.

[25] 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, Sep. 2015.

[26] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," in *ACM International Conference Proceeding Series*, 2010, p. 1.