Transfer Learning for Cross-Project Change-Proneness Prediction in Object-Oriented Software Systems: A Feasibility Analysis

Lov Kumar
NIT Rourkela (India)
lovkumar505@gmail.com

Ranjan Kumar Behera
NIT Rourkela (India)
jranjanb.19@gmail.com

Santanu Rath
NIT Rourkela (India)
skrath@nitrkl.ac.in

Ashish Sureka
ABB (India)
ashish.sureka@in.abb.com

DOI: 10.1145/3127360.3127368
http://doi.acm.org/10.1145/3127360.3127368

ABSTRACT

Change-prone classes or modules are defined as regions of the source code which are more likely to change as a result of a software development of maintenance activity. Automatic identification of change-prone classes are useful for the software development team as they can focus their testing efforts on areas within the source code which are more likely to change. Several machine learning techniques have been proposed for predicting change-prone classes based on the application of source code metrics as indicators. However, most of the work has focused on within-project training and model building. There are several real word scenario in which sufficient training dataset is not available for model building such as in the case of a new project. Cross-project prediction is an approach which consists of training a model from dataset belonging to one project and testing it on dataset belonging to a different project. Cross-project change-proneness prediction is relatively unexplored.

We propose a machine learning based approach for cross-project change-proneness prediction. We conduct experiments on 10 open-source Eclipse plug-ins and demonstrate the effectiveness of our approach. We frame several research questions comparing the performance of within project and cross project prediction and also propose a Genetic Algorithm (GA) based approach for identifying the best set of source code metrics. We conclude that for with-in project experimental setting, Random Forest (RF) technique results in the best precision. In case of cross-project change-proneness prediction, our analysis reveals that the NDTF ensemble method performs higher than other individual classifiers (such as decision tree and logistic regression) and ensemble methods in the experimental dataset. We conduct a comparison of within-project, cross-project without GA and cross-project with GA and our analysis reveals that cross-project with GA performs best followed by within-project and then cross-project without GA.

Keywords

Software Engineering, Source Code Metrics, Genetic Algorithm, Classification Techniques, Object-Oriented Software Systems, Cross-Project Prediction, Machine Learning, Source Code Analysis, Predictive Modeling, Change-Proneness Prediction

1. RESEARCH MOTIVATION AND AIM

Identifying change-prone classes within an object oriented software system is important for the purpose of optimally allocating development and testing resources. Approaches for predicting change-prone classes is an area that has attracted a lot of research attention [9][11][13][14][16]. Several researchers have proposed machine learning techniques using source code metrics based features for change-prone class prediction and conducted experiments on a wide variety of open source and closed source software projects [9] [11] [13] [14] [16]. Within project change prone prediction consists of training and applying the predictive model on the same project (source and target projects are same). One of the limitations of within project prediction is that a sufficiently large amount of training data is required for model building. However in many situations sufficient amount of training data is not available to train a model such as in the case of a new project or a product going through initial releases. Cross project prediction consists of using data from other projects of similar characteristics for model building and then applying the predictive model for identifying the change-prone classes (source project and target projects are different) [1][12][17]. Cross project change prone prediction is a relatively unexplored area with very few empirical studies and evidences. The motivation of the study presented in this paper is to investigate the effectiveness of cross-project change prone prediction by conducting experiments on several open source software projects. The specific research aims of the work presented in this paper are the following:

1. To investigate the feasibility and performance of machine-learning based cross-project change-proneness prediction by conducting experiments on open-source object-oriented Java applications.
2. To investigate and compare the performance of several machine learning classifiers, ensemble methods and source code metrics for the task of cross-project change-proneness prediction. To compare the results of with-in project and cross-project prediction.
3. To conduct an examination of the source code metrics and project combination and identify correlations which provides guidance on identifying the best combination of training data.
4. To apply Genetic Algorithm (GA) for identifying the best set of source code metrics and project for cross-project change-proneness prediction

2. RELATED WORK

In this Section, we present previous work related to the study presented in this paper. There are two lines of research which are related to our study. One research direction is on the application of source code metrics and machine learning techniques

for software change prediction. The other line of research is on cross-project fault or defect prediction.

2.1 Software Change-Proneness Prediction

Zhu et al. conduct experiment on an open-source software product called as Datacrow and build a solution to automatically identify change-prone classes [16]. Their experimental analysis reveals that approximately 80% of the lines changed in the system are located within 20% of the classes (validation of the Pareto's Law) [16]. Kumar et al. compute source code metrics for Programmable Logic Controller (PLC) programming languages and study their correlation with change proneness at code tab and program organization unit level [11]. Their study is on change proneness in the domain of industrial automation engineering [11][8]. Kumar et al. use 62 software metrics (inheritance, size, cohesion and coupling) to develop a change-proneness model [9]. They use eight different machine learning algorithms, five feature selection techniques and two ensemble methods [9]. Kumar et al. conduct a case-study using data from eBay web services for predicting change prone web-services [10]. Malhotra et al. perform a study on six open source project dataset and focus on building effective change-proneness prediction model in the case of imbalanced dataset [14]. They experiment with several sampling methods and their finding reveals that resample with replacement sampling method is effective [14]. Eski et al. perform correlation analysis between changes in a software system and object oriented metrics [4]. Yan et al. use a discriminative Probability Latent Semantic Analysis (DPLSA) model and evaluate their approach on five open source projects for change classification [15]. Malhotra et al. conduct experiments on cross-project change prediction and validate their approach on three open source Apache projects: Abdera, POI and Rave [13].

2.2 Cross-Project Defect Prediction

Zimmermann et al. conduct experiments on cross-project defect prediction consisting of 12 real-world applications and execution of 622 cross-project predictions [17]. He et al. investigate whether cross-project defect prediction is feasible by performing a series of experiments on 34 datasets derived from 10 open source software (OSS) projects [6]. Ma et al. present a study on transfer learning in which they build cross-company defect prediction model [12]. In their experiments the source and target data belong to projects from different companies [12]. Herbold et al. propose distance-based strategies for training data selection for cross-project defect prediction [7]. Canfora et al. use a multi-objective logistic regression model developed using evolutionary approaches such as genetic algorithm and propose a multi-objective approach for cross-project defect prediction [1]. Fukushima et al. conduct an empirical study on just-in-time cross-project defect prediction model [5]. They conduct a study on 11 open-source projects and also apply ensemble techniques [5]. Chen et al. examine cross-company defect prediction [2] and propose an algorithm called as Double Transfer Boosting (DTB) to enhance the performance of cross company defect prediction by reducing negative samples in cross company data [2].

3. RESEARCH CONTRIBUTIONS

In context to the state-of-the-art and existing work, the work presented in this paper makes several unique and novel research contributions:

1. We conduct an in-depth empirical analysis on cross-project change-proneness prediction on ten open-source software plug-ins for Eclipse using eight different machine learning classifiers (including the ensemble methods). The experiments are conducted for both within project and cross-project cases. We present empirical results on comparison of the different classifiers.

2. We conduct an investigation on the characteristics of the source code metrics and identifying good predictors influencing the selection of training data for cross-project change-proneness prediction. We provide empirical evidences and results regarding identification of pair of source code metrics and project dataset which perform higher for the change-proneness prediction task for a project.

3. We propose an approach to identify the best set of source code metrics and project for cross-project change-proneness prediction using Genetic Algorithms (GA)

4. EXPERIMENTAL DATASET DESCRIPTION

We conduct experiments on open-source publicly available dataset so that our experiments can be easily reproduced and the results can be replicated. Table 1 displays a list of 10 Eclipse 2.0 and Eclipse 2.1 plug-ins used in our experiments. The source code for the Eclipse source build 2.0 is downloaded from the URL[1] as a zip file and similarly the source code for the the Eclipse source build 2.0 is downloaded from the URL[2] as a zip file. All the 10 plug-ins are Java based applications and are present in both the versions of the Eclipse. We compute the object oriented metrics, the number of classes and the number of source lines of code changes for each Java class in Eclipse 2.0 version which also appears in Eclipse 2.1 version.

The source code metrics are computed using a tool called as Understand from Scitools[3]. We compute 61 metrics listed on the website[4] of the Understand tool. The metrics cover a diverse range of source code properties such as lines of code, cyclomatic complexity, coupling between objects, class methods, class variables, functions, instance methods and variables and depth of inheritance tree. We use a diff tool called as Jar Compare[5] for computing differences between Class files in Java JAR archives. The tool is used for comparing changes between software builds or releases and we use it to identify the Classes which were changes between the two versions within a plug-in in our experimental dataset. Table 1 reveals that the largest plug-in in our dataset is JDT having 1943 classes in Eclipse version 2.0 and number of changed classes as 1221. The number of changed classes vary from a minimum of 31 to a maximum of 1221. The percentage of changed classes vary from 29.81% to a maximum of 67.67%.

Table 1: Experimental Dataset Details

Proj ID	Name	# class	# changed classes	% changed classes
DS1	compare	83	38	45.78
DS2	webdav	104	31	29.81
DS3	debug	133	90	67.67
DS4	update	249	167	67.07
DS5	core	250	90	36.00
DS6	swt	344	126	36.63
DS7	team	372	236	63.44
DS8	pde	487	269	55.24
DS9	ui	826	516	62.47
DS10	jdt	1943	1221	62.84

[1] https://goo.gl/bFM30Y
[2] https://goo.gl/3z33Rw
[3] https://scitools.com/
[4] https://scitools.com/support/metrics_list/
[5] http://www.extradata.com/products/jarc/

Table 2: Experimental Performance Results for *with-in* Change-Proneness Prediction

Project	LOGR		ANN		RBFN		DT		RF		BTE		MVE		NDTF	
	Acc	FME	Acc	FME	Acc	FME	Acc	FME	Acc	FME	Acc	FME	Acc	FME	Acc	FME
compare	63.86	0.69	78.31	0.80	73.49	0.78	63.86	0.66	73.49	0.77	77.11	0.79	81.93	0.83	72.29	0.73
webdav	68.27	0.76	76.92	0.84	83.65	0.89	73.08	0.82	75.96	0.83	82.69	0.88	83.65	0.89	77.88	0.85
debug	61.65	0.45	77.44	0.63	75.94	0.60	71.43	0.50	68.42	0.54	73.68	0.59	76.69	0.62	78.20	0.60
update	70.68	0.58	77.51	0.65	76.71	0.61	72.69	0.51	78.31	0.67	79.92	0.67	80.72	0.69	74.30	0.52
core	69.60	0.76	74.00	0.81	72.00	0.79	70.80	0.78	74.00	0.81	79.20	0.85	74.80	0.82	71.20	0.79
swt	77.33	0.83	80.23	0.85	83.14	0.87	77.33	0.82	78.49	0.84	78.78	0.84	81.98	0.86	79.94	0.84
team	67.20	0.53	71.24	0.57	68.28	0.52	66.40	0.48	68.82	0.53	72.04	0.57	70.16	0.54	69.35	0.51
pde	71.46	0.69	74.54	0.73	76.39	0.74	68.99	0.66	72.28	0.71	74.95	0.73	73.92	0.71	72.48	0.70
ui	72.15	0.62	73.37	0.62	74.82	0.66	68.16	0.52	73.61	0.65	72.40	0.60	76.39	0.68	72.76	0.63
jdt	74.94	0.66	76.89	0.67	76.84	0.67	69.99	0.54	75.30	0.67	75.30	0.63	78.28	0.70	72.31	0.59

5. EXPERIMENTAL ANALYSIS AND RESULTS

We structure our study in the form of Research Questions (RQ). We frame six RQs addressing the research aims and contributions defined in the previous section.

RQ 1: What is the performance of the proposed change-proneness prediction models in the case of with-in project training and testing?

The primary objective of experiments is to investigate the application of proposed model to predict change-proneness of future releases and unseen similar natured object-oriented software. In this paper, two different validation methods have been considered to validate the developed change-proneness models. In *with-in* validation method, k-fold cross validation (k = 10 in our case) has been considered to validate the developed models. Cross-validation is a well-known technique used in statistical learning to assess the generalizability of predictive models and also compare the predictive models built from a dataset. It works by partitioning the data into two sets or portions: training and testing. One of the portions of the divided set is used to train or learn the model and the rest of the data is used to validate the model built during training. The significance and reason of applying cross-validation is the usage of dataset for both training and testing for the purpose of model building and evaluation. The performance of each model on each fold of the k-fold can be tracked using predetermined information retrieval performance metrics such as accuracy, precision, recall and f-measure.

Table 2 displays the performance parameters value of with-in validation method. The machine learning techniques used by us are (mentioned in Table 2): LOGR (Logistic Regression), ANN (Artificial Neural Networks), RBFN (Radial Basis Function Network), DT (Decision Trees), RF (Random Forests), BTE (Best Training Ensemble), MVE (Majority Voting Ensemble) and NDTF (Non Linear Decision Tree Forest). Zimmermann et al. [17] defined good predictors as those with an accuracy value greater than 75%. From Table 2, we observe that the models developed using machine learning techniques results in an accuracy value varying from 70% to 80%. The experimental results provides encouraging results on the predictive capability of machine learning techniques for developing change-proneness prediction models.

Figure 1 shows the box-plot diagrams for all four performance parameters such as accuracy, precision, recall, and F-Measure. Every diagram contains eight box-plots: one for each classification technique. The box-plot diagrams help us to understand the performance of all the machine learning methods within a single diagram. The line in the middle of each box represents the median value. The predictive model having a high median value is a better change-proneness prediction model. We draw the following inferences from the box plots in Figure 1:

1. The precision box-plots show that the average precision value for all dataset are greater than 0.5, with random forest (RF) technique showing best precision value.

2. Similarly, the recall box-plots show that the average recall value of artificial neural network (ANN), radial basic neural network (RBFN), and three ensemble methods such as BTE, MVE, NDTF are greater than 0.7, with major voting ensemble method showing best recall value.

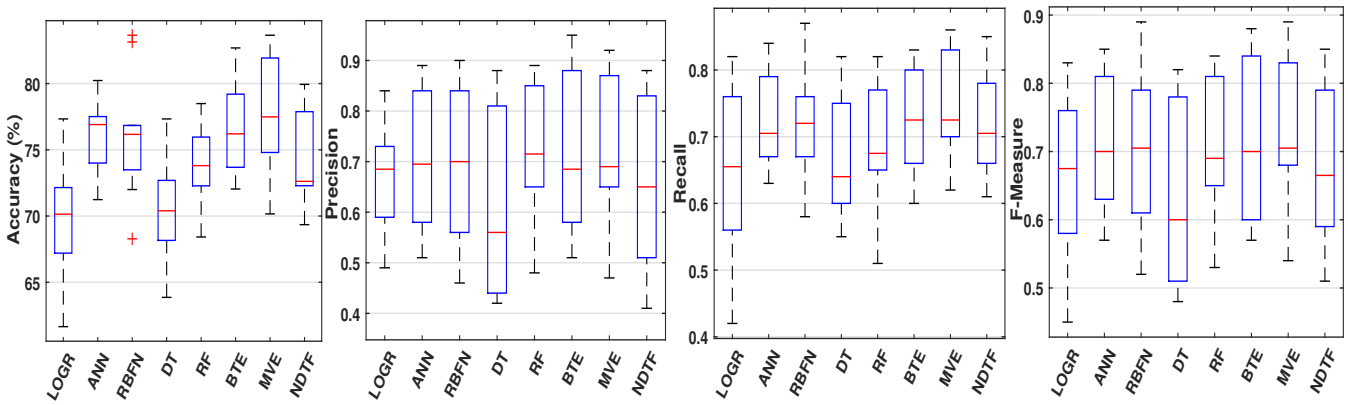


Figure 1: *With-in* Project Change-Proneness Prediction Model Performance Displayed using Box-Plots

3. One of the most widely used performance parameter in data mining is F-measure. F-measure integrate both precision and recall value in a single indicator. The model which has a high F-Measure value is the best change-proneness prediction model. From these diagrams, it can be seen that the change-proneness model developed using major voting ensemble method results in high values of F-Measure as compared to a model developed using other techniques.

In our study, we also consider Wilcoxon signed rank test to compare different classifiers with an objective to determine whether the developed change-proneness models developed using different classifiers perform equally well or are their noticeable differenences in their performance. In our experiment, Wilcoxon test with Bonferroni correction is considered for comparison analysis [3].

Eight different classifiers are used to developed change-proneness model. Each classifier is applied on ten different dataset. Therefore for each technique a total number of four sets (one for each performance) is used, each with 10 data points (10 datasets). The results of the pair-wise comparisons of different classifiers for accuracy performance parameter are shown in Table 3. The results for other parameters are of similar types. In this work, eight classifiers have been considered for analysis, i.e. a total number of twenty eight (28) different pairs are possible (${}^{8technique}C_2 = 8*7/2 = 28$). All results are analyzed at a 0.05 significance level. Hence, the null hypothesis that "there is no significant difference between the two classifiers" is rejected only if $p - value \leq \frac{0.05}{28} = 0.0018$. From Table 3, we observe that out of 28 pairs of training methods, 10 are found to have significant results. From Table 3, it is also observed that the MVE yields better results compared to other classifiers based on the value of performance mean difference.

RQ 2: What is the performance of the proposed change-proneness prediction models in the case of *cross-project* validation?

In cross-project validation method, the training data is obtained from other projects. In this method, all combinations of datasets of other projects are used to train the model. We test all possible combinations and select those combination which can provide the best prediction result. Therefore for each project, a total of 4088 ((${}^9C_1+{}^9C_2+{}^9C_3+{}^9C_4+{}^9C_5+{}^9C_6+{}^9C_7+{}^9C_8+{}^9C_9$ combination) * 8 classification techniques = (9 + 36 + 84 + 126 + 126 + 84 + 36 + 9 + 1)*8=511*8) distinct classification models are built. Figure 2 shows the step-by-step procedure of generating training data and finding best training set for a project. For example, for the given project *compare*, we use the combinations of data sets outside the *compare* project as training data (e.g., webdav, debug, update, core, swt, team, pde, ui, jdt). If the <debug, update> combination as a training data produce best result for *compare* project then we can say that <debug, update> is the most suitable training data for "Compare" project in the cross-project context. Figure 3 shows the box-plot diagrams for three performance parameters such as accuracy, precision, and recall. The box-plot diagrams for F-Measure performance parameter is of similar type. Every diagram contains eight box-plots:

1. 0->1: With-in validation method (cross-validation method).
2. 1->1: One other project for training data and one project for testing data.
3. 2->1: Combination of two other project as a training data and one project for testing data.

Table 3: Wilcoxon Signed Rank Test of Different Classifiers for the Case of *With-in* Project Validation Method

p-value								
	LOGR	**ANN**	**RBFN**	**DT**	**RF**	**BTE**	**MVE**	**NDTF**
LOGR	1.00	0.00	0.00	0.95	0.00	0.00	0.00	0.03
ANN	0.00	1.00	0.85	0.00	0.02	0.92	0.04	0.04
RBFN	0.00	0.85	1.00	0.00	0.16	0.77	0.10	0.04
DT	0.95	0.00	0.00	1.00	0.01	0.00	0.00	0.00
RF	0.00	0.02	0.16	0.01	1.00	0.01	0.00	0.77
BTE	0.00	0.92	0.77	0.00	0.01	1.00	0.28	0.06
MVE	0.00	0.04	0.10	0.00	0.00	0.28	1.00	0.01
NDTF	0.03	0.04	0.04	0.00	0.77	0.06	0.01	1.00
Mean Difference								
	LOGR	**ANN**	**RBFN**	**DT**	**RF**	**BTE**	**MVE**	**NDTF**
LOGR	0.00	-6.33	-6.41	-0.56	-4.15	-6.89	-8.14	-4.36
ANN	6.33	0.00	-0.08	5.77	2.18	-0.56	-1.81	1.97
RBFN	6.41	0.08	0.00	5.85	2.26	-0.48	-1.73	2.05
DT	0.56	-5.77	-5.85	0.00	-3.60	-6.33	-7.58	-3.80
RF	4.15	-2.18	-2.26	3.60	0.00	-2.74	-3.99	-0.20
BTE	6.89	0.56	0.48	6.33	2.74	0.00	-1.25	2.53
MVE	8.14	1.81	1.73	7.58	3.99	1.25	0.00	3.78
NDTF	4.36	-1.97	-2.05	3.80	0.20	-2.53	-3.78	0.00

4. 3->1: Combination of three other project as a training data and one project for testing data.
5. 4->1: Combination of four other project as a training data and one project for testing data.
6. 5->1: Combination of five other project as a training data and one project for testing data.
7. 6->1: Combination of six other project as a training data and one project for testing data.
8. 7->1: Combination of seven other project as a training data and one project for testing data.
9. 8->1: Combination of eight other project as a training data and one project for testing data.
10. 9->1: Combination of all nine other project as a training data and one project for testing data.

From Figure 3, it can be seen that the with-in validation method yields better results as compare to cross-project validation method. From Figure 3, it is also observed that the change-proneness prediction model trained using combination of all nine other project as a training data produce better results as compare to others. Figure 4 shows the box-plot diagrams for three performance parameters such as accuracy, precision, and recall. The box-plot diagrams for F-Measure performance parameter is of similar type. Every diagram contains eight box-plots: one for each classification technique. From Figure 4, it can be seen that the change-proneness model developed using NDTF ensemble method results in high value of performance parameters as compared to predictive models developed using other techniques.

5.1 Comparison of Predictive Models

In our experiments, Wilcoxon test with Bonferroni correction has been considered for conducting comparison between the predictive models.

5.1.1 Classification Techniques

Eight different classifiers are used to develop change-proneness model. Each classifier is applied on ten different dataset. Hence, for each technique, a total number of four sets (one for each performance) is used, each with 5110 data points. Combination:

10 datasets * $({}^{9}C_1+{}^{9}C_2+{}^{9}C_3+{}^{9}C_4+{}^{9}C_5+{}^{9}C_6+{}^{9}C_7+{}^{9}C_8+{}^{9}C_9)$

The results of the pair-wise comparisons of different classifiers for accuracy performance parameter are shown in Table 4a. The results for other parameters are of similar types. In our study, eight classifiers have been considered for analysis, i.e. total number of twenty eight (28) different pairs are possible $({}^{8technique}C_2 = 8 * 7/2 = 28)$ and all results are analyzed at a 0.05 significance level. Hence, the null hypothesis "there is no significant difference between the two classifiers" is rejected only if $p - value \leq \frac{0.05}{28} = 0.0018$. From Table 4a, it can be seen that, out of 28 pairs of training methods, 24 are found to have significant results. From Table 4a, it is also observed that the NDTF yields better results compared to other classifiers based on the performance value of mean difference.

5.1.2 *Different Combinations of Training Dataset*

Ten different combination length of training datasets are used to train the change-proneness models. Each combination set has different data points i.e., with-in validation method (0->1) has 80 points (10 projects * 8 classifiers), 2->1 cross project validation methods having 2880 data points ($^{9}C_2$ combinations *10 projects * 8 classifiers) etc.. The results of the pair-wise comparisons of different combinations for accuracy performance parameter are shown in Table 4b. The results for other parameters are of similar types. In this work, ten different combinations have been considered for analysis, i.e. total number of forty five (45) different pairs are possible $({}^{10technique}C_2 = 10 * 9/2 = 45)$ and all results are analyzed at a 0.05 significance level. Hence, the null hypothesis "there is no significant difference between the two combinations" is rejected only if $P - value \leq \frac{0.05}{45} = 0.0011$. From Table 4b, it can be seen that, out of 45 pairs of training methods, 27 are found to have significant results. From Table 4b, it is also observed that the with-in validation method yields better results compared to cross-project validation method. Further, it is also observed that the 9− > 1 combination yields better result as compare to other combination in cross-project validation method.

RQ 3: Can cross-project validation method provide acceptable prediction results (based on acceptable prediction result criteria in literature)?

Table 4: Wilcoxon Test with Bonferroni Correction

(a) Wilcoxon Test: Among different Classifier

p-value								
	LOGR	**ANN**	**RBFN**	**DT**	**RF**	**BTE**	**MVE**	**NDTF**
LOGR	1.00	0.44	0.00	0.00	0.00	0.33	0.00	0.00
ANN	0.44	1.00	0.00	0.00	0.00	0.07	0.00	0.00
RBFN	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
DT	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
RF	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.74
BTE	0.33	0.07	0.00	0.00	0.00	1.00	0.00	0.00
MVE	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
NDTF	0.00	0.00	0.00	0.00	0.74	0.00	0.00	1.00
Mean Difference								
LOGR	0.00	-0.17	1.09	4.26	-2.93	0.23	-1.17	-2.85
ANN	0.17	0.00	1.27	4.43	-2.75	0.40	-1.00	-2.67
RBFN	-1.09	-1.27	0.00	3.16	-4.02	-0.87	-2.27	-3.94
DT	-4.26	-4.43	-3.16	0.00	-7.18	-4.03	-5.43	-7.10
RF	2.93	2.75	4.02	7.18	0.00	3.15	1.75	-0.08
BTE	-0.23	-0.40	0.87	4.03	-3.15	0.00	-1.40	-3.08
MVE	1.17	1.00	2.27	5.43	-1.75	1.40	0.00	-1.67
NDTF	2.85	2.67	3.94	7.10	0.08	3.08	1.67	0.00

(b) Wilcoxon test: Different Combination

p-value										
	WIP	**1-1**	**2-1**	**3-1**	**4-1**	**5-1**	**6-1**	**7-1**	**8-1**	**9-1**
WIP	1	0	0	0	0	0	0	0	0	0
1-1	0	1	0.001	0.008	0	0	0	0	0.995	0
2-1	0	0.001	1	0.225	0	0	0	0	0.001	0
3-1	0	0.008	0.225	1	0	0	0	0	0.008	0
4-1	0	0	0	0	1	0.949	0.005	0.012	0	0
5-1	0	0	0	0	0.949	1	0.006	0.01	0	0
6-1	0	0	0	0	0.005	0.006	1	0	0	0
7-1	0	0	0	0	0.012	0.01	0	1	0	0
8-1	0	0.995	0.001	0.008	0	0	0	0	1	0
9-1	0	0	0	0	0	0	0	0	0	1
Mean Difference										
	WIP	**1-1**	**2-1**	**3-1**	**4-1**	**5-1**	**6-1**	**7-1**	**8-1**	**9-1**
WIP	0.00	17.48	19.02	18.71	21.08	21.09	21.60	20.44	17.48	11.17
1-1	-17.48	0.00	1.54	1.22	3.59	3.61	4.12	2.95	0.00	-6.31
2-1	-19.02	-1.54	0.00	-0.31	2.06	2.07	2.58	1.42	-1.54	-7.85
3-1	-18.71	-1.22	0.31	0.00	2.37	2.38	2.90	1.73	-1.23	-7.53
4-1	-21.08	-3.59	-2.06	-2.37	0.00	0.01	0.53	-0.64	-3.60	-9.90
5-1	-21.09	-3.61	-2.07	-2.38	-0.01	0.00	0.52	-0.65	-3.61	-9.92
6-1	-21.60	-4.12	-2.58	-2.90	-0.53	-0.52	0.00	-1.17	-4.13	-10.43
7-1	-20.44	-2.95	-1.42	-1.73	0.64	0.65	1.17	0.00	-2.96	-9.26
8-1	-17.48	0.00	1.54	1.23	3.60	3.61	4.13	2.96	0.00	-6.31
9-1	-11.17	6.31	7.85	7.53	9.90	9.92	10.43	9.26	6.31	0.00

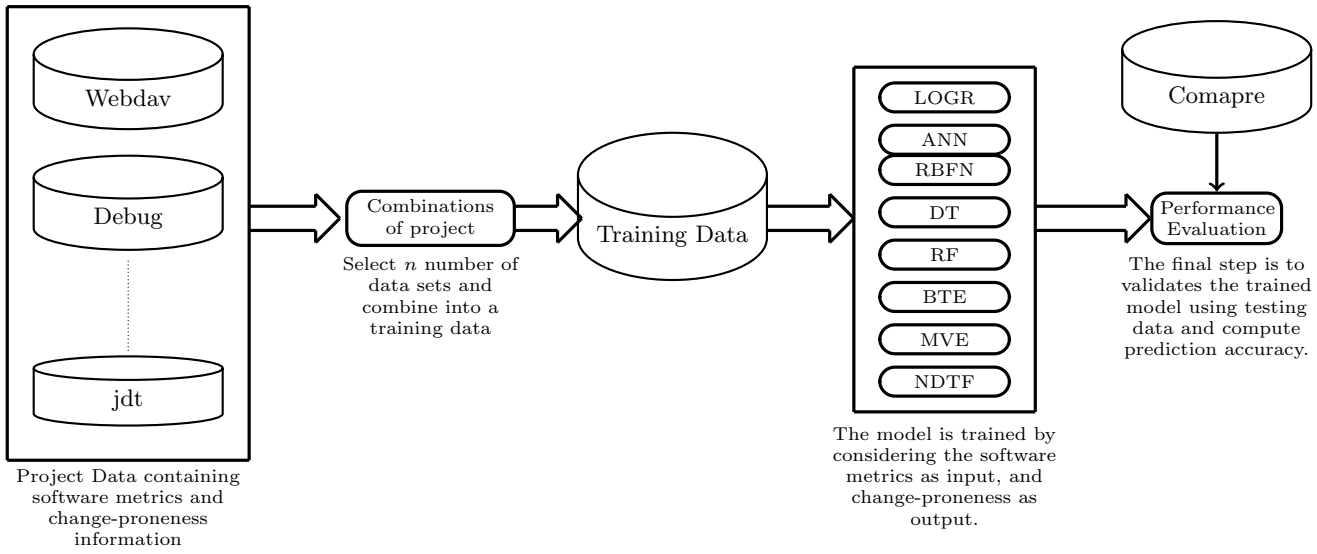


Figure 2: Framework of *Cross-project* Validation Method for *Compare* Project Data

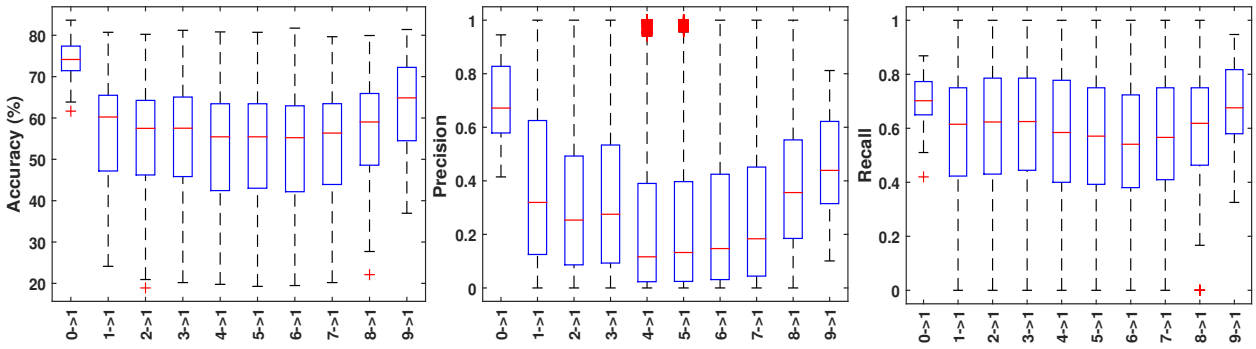


Figure 3: Cross-Project Change-Proneness Prediction Model Performance

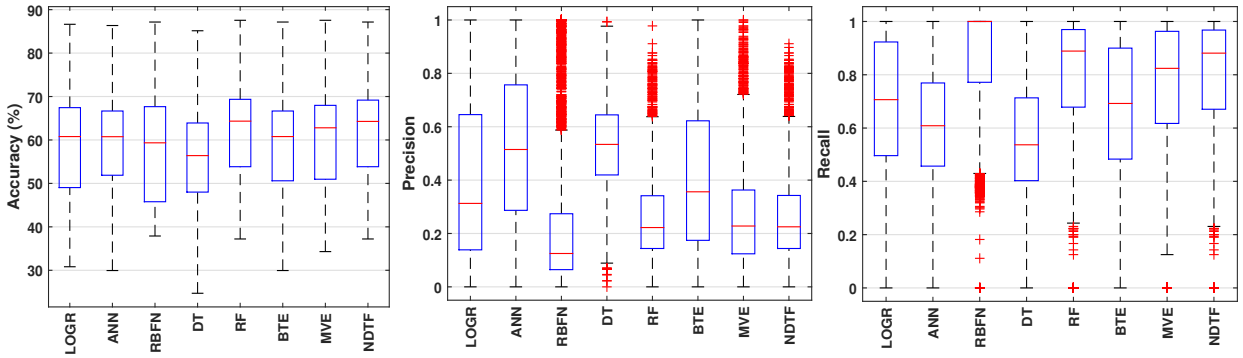


Figure 4: Cross-Project Change-Proneness Prediction Model Performance (Using Classification Techniques)

The objective of this work is to precisely detect change-proneness classes of object-oriented software. These change-proneness classes are predicted using developed prediction models. In this work, four different performance parameters are used to compute the predictive power of developed change-proneness models. Zhimin et al. [6] defined good predictors as those with Recall greater than 0.70 and Precision greater than 0.50. The developed models with Recall $\geq$ 0.70 and Precision $\geq$ 0.50 as acceptable results. Our objective in this research question is to verify whether the trained model using cross-project datasets provide acceptable results or not.

In this paper, for each project, we select training data from 9 data sets of other projects. Hence, for each projects, a total of 4088:

(($^9C_1$+$^9C_2$+$^9C_3$+$^9C_4$+$^9C_5$+$^9C_6$+$^9C_7$+$^9C_8$+$^9C_9$ combination) * 8

classification techniques = (9 + 36 + 84 + 126 + 126 + 84 + 36 + 9 + 1)*8=511*8) distinct predictor (NOP) are built in the study. Table 5 shows the results of cross-project validation method of each projects. Table 5 also displays information about the most suitable training data and prediction results. From Table 5, it can be seen that the ratios of successful cross-project change-proneness prediction range from 0.02% to 15.53%. From Table 5, it is also observed that the cross-project validation methods produce 635 number of good prediction (NOGP) with ratio of 15.53 (%GP) in case of "swt" project.

Similarly, Table 6 shows the results of cross-project validation method of each classifier. Each classifier is applied on ten different datasets, so each classifier results in 5110 different predictor

Table 5: Cross-Project Prediction

				Most suitable training data and prediction result				
	NOP	NOGP	%GP	Acc	PRE	REC	FME	Best Train Data
compare	4088	71	1.74	80.72	0.91	0.77	0.84	DS2+DS8
webdav	4088	169	4.13	81.73	0.93	0.83	0.88	DS5+DS6+DS7
debug	4088	8	0.20	80.45	0.56	0.77	0.65	DS1
update	4088	234	5.72	87.15	0.72	0.87	0.79	DS1+DS2
core	4088	161	3.94	80.80	0.97	0.78	0.87	DS1+DS2+DS8
swt	4088	635	15.53	86.34	0.95	0.85	0.90	DS2+DS4+DS9
team	4088	1	0.02	74.46	0.51	0.71	0.60	DS1
pde	4088	17	0.42	73.51	0.71	0.70	0.71	DS1
ui	4088	32	0.78	75.54	0.59	0.71	0.65	DS4
jdt	4088	9	0.22	75.81	0.61	0.70	0.65	DS1

(NOP) (10 datasets *

($^9C_1$+$^9C_2$+$^9C_3$+$^9C_4$+$^9C_5$+$^9C_6$+$^9C_7$+$^9C_8$+$^9C_9$ combination)).

From Table 5, it can be seen that the ratio of successful cross-project change-proneness prediction ranges from 1.23% to 5.36%. Table 5 reveals that the model developed using artificial neural network with cross-project validation methods produces 274 number of good prediction (NOGP) with ratio of 5.36 (%GP).

In this paper, for each project, a total number of 2048 (($\frac{1 + {}^9C_1 + {}^9C_2 + {}^9C_3 + {}^9C_4 + {}^9C_5 + {}^9C_6 + {}^9C_7 + {}^9C_8 + {}^9C_9}{2}$)* 8 classification techniques

$= \frac{1 + 9 + 36 + 84 + 126 + 126 + 84 + 36 + 9 + 1}{2}$ * 8= 256*8)

times of one project data is used for training purpose i.e., for

Table 6: Cross-Project Prediction (Measuring the Performance of Classification Technique)

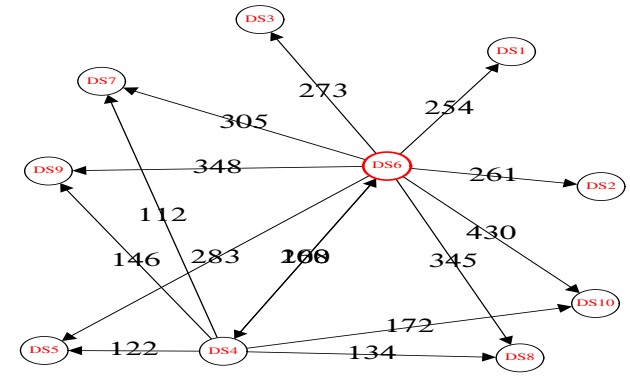
	NOP	NOGP	%GP	Acc	PRE	REC	FME
LOGR	5110	213	4.17	80.81	0.85	0.85	0.85
ANN	5110	274	5.36	80.72	0.02	0.50	0.04
RBFN	5110	63	1.23	81.73	0.96	0.77	0.85
DT	5110	167	3.27	78.49	0.18	0.56	0.28
RF	5110	145	2.84	81.20	0.19	0.20	0.19
BTE	5110	189	3.70	79.12	0.38	0.66	0.48
MVE	5110	140	2.74	81.40	0.76	0.89	0.82
NDTF	5110	146	2.86	80.45	0.19	0.20	0.19

project DS1, DS2 is used 2048 as a training data. Figure 5 shows the good predictor graph for each pair of dataset. From Figure 5, it can be seen that the DS6 to DS10 is the best pair training and testing data i.e., the model trained by considering DS6 produce 430 number of good prediction (NOGP) out of 2048 predictor in case of DS10. Figure 5 consists of three graphs. Each graph has nodes and edges. The nodes represents one of the ten projects in our dataset. The edges are directed and represents the source and target project relationship. The number of label on the edge represents the number of times the models trained by the source project has resulted in good predictors for the target project. As shown in Figure 5, the density of the graph is calibrated by filtering on the number of good predictors. For example, the number of good predictors is 134 when the source project is DS4 and the target project is DS8.

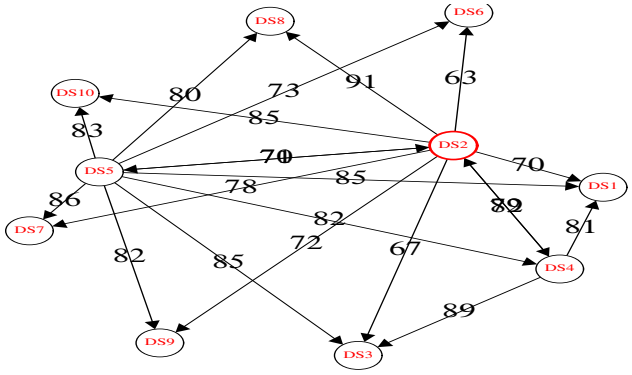
RQ 4: Are the characteristics of source code metrics valuable to take decision regarding selection of suitable training data for cross-project change-proneness prediction?

From Tables 5 and 6, we observe that the cross-project change-proneness predictions works in few cases (cases where the %GP is very low). A cross-project change-proneness predictions is considered to be good if its recall value is greater than 0.70 and the precision value is greater than 0.50. In this section, we present a verification analysis on the relationship and association between the distributional characteristics of input source code metrics and output prediction results. In this work, initially we have generated *train-test-result* instance for each project. This instance contain three parts: distributional characteristics of the training data set, distributional characteristics of the test data set, and the prediction result. The prediction results here are discretized into two groups: prediction results meeting the criteria Recall $\geq$ 0.70 and Precision $\geq$ 0.50 are labeled as "yes" while the others are labeled as "no". In this work, ten different characteristics such as Minimum (Min), Maximum (Max), Mean, Median, Standard Deviation (std), Variance (var), Skewness, Kurtosis, First Quartile, and Third Quartile are used to describe the distribution of source code metrics. After finding characteristics of each source code metrics of training and testing datasets, these characteristics are combined together, which results in a set of 1220 features (i.e., 61 source code metrics *10 characteristics for training data + 61 source code metrics *10 characteristics for testing data). These 1220 features are used to to describe the distributional characteristics of training and testing for each instance. The last attribute (attribute no: 1221) indicating whether the prediction is successful. Figure 6 shows the sequence of operations for generating a train-test-result instance from a cross-project change-proneness prediction.

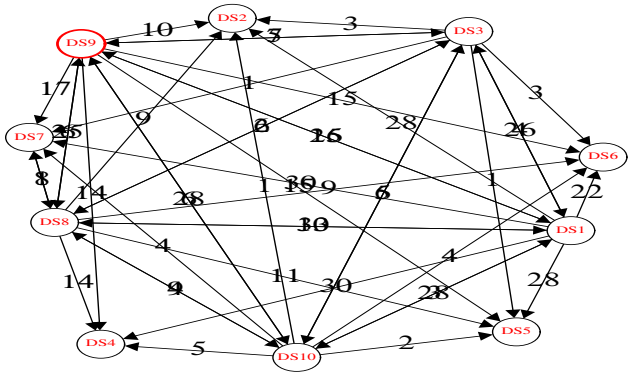
We generate 40880 train-test-result instances from predictions (511 combinations * 10 data sets * 8 prediction techniques). We



(a) No of Good Predictor>100



(b) 50<No of Good Predictor<=100



(c) No of Predictor<=50

Figure 5: Good Predictor Graph

have followed the steps proposed by Zimmermann et al. [17], using decision tree to investigate whether prediction results of cross-project change-proneness predictions are related to distributional characteristics of source code metrics of training sets and test sets. If the decision tree classifier can automatically determine those successful cross-project change-proneness predictions precisely, then it implies that the distributional characteristics of data set are related to prediction results within the cross-project scenario. In this work, 10-fold cross validation has been considered to validate the models.

The generated decision tree using decision tree technique by considering distributional characteristics of train and test data as input contain 466 leaf node, among which 253 leaf nodes are labeled as "yes" and rest 213 leaf nodes are labeled as "No" i.e., de-

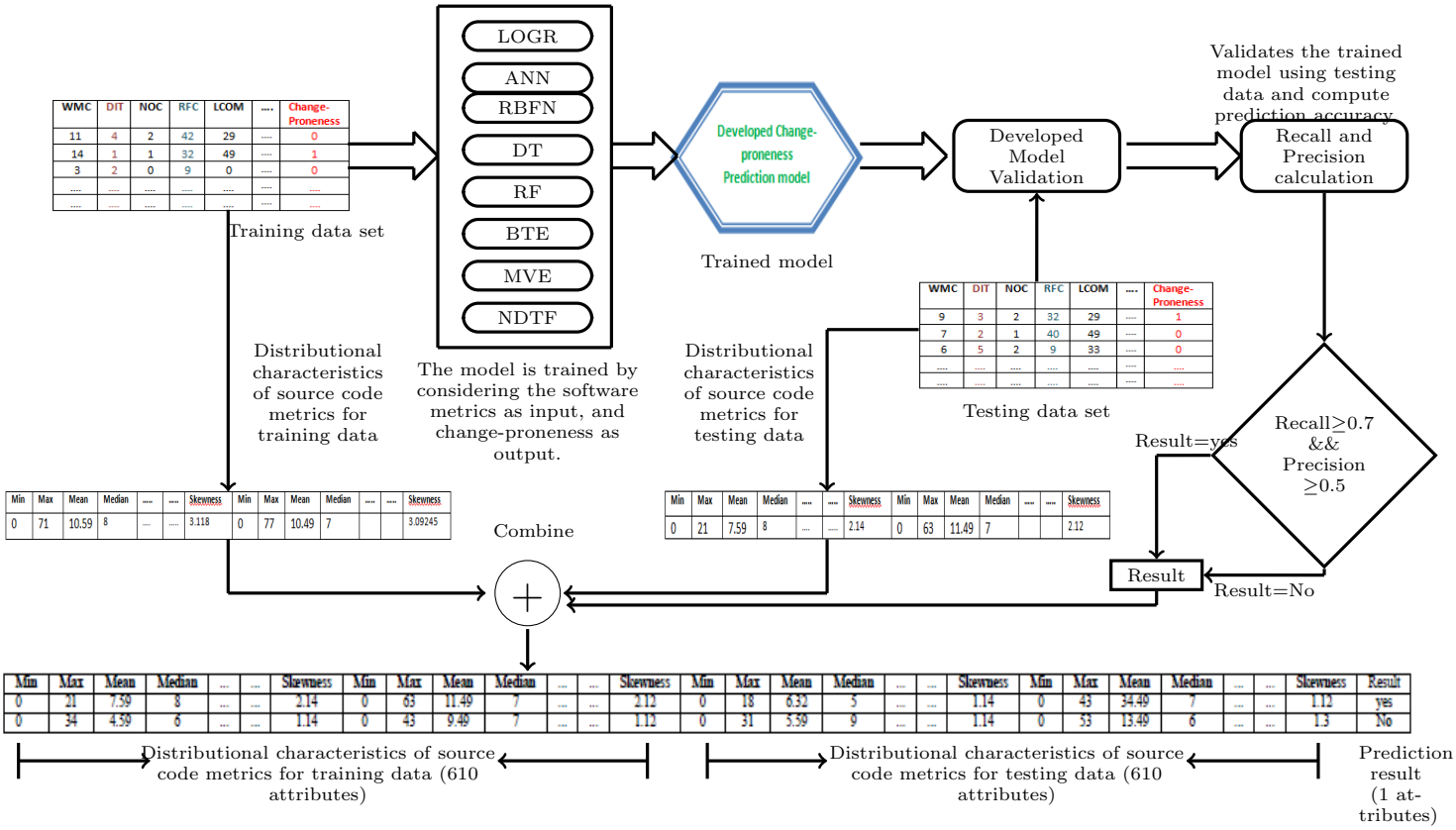


Figure 6: Procedure of Generating a Train-Test Result Instance from a Cross-Project Change-Proneness Prediction

cision tree contain 253 rules for detecting successful cross-project change-proneness predictions. The top two rules generated using decision tree for each category are shown in Table 7. The second column of Table 7 shows the number of predictions satisfying the rule. The high value of support shows the characteristics of source code metrics valuable to take decision regarding selection of suitable training data for cross-project change-proneness prediction.

Table 7: Rules Induced by Decision Tree Classifier for Detecting Successful Cross-Project Change-Proneness Predictions

Rule	**No of Support**
if Max(NA-test) <776.5 and Skewness(DCMEC-test) <10.1735 and Skewness(TCC-train) <0.317803 and Mean(Co-train) then Result= yes	7175
if Max(NA-test) <776.5 and Skewness(DCMEC-test) >=10.1735 and Skewness(NA-test) <4.10542 and Mean(NewLCOM5-train) >=2856.21 then Result= yes	1809
if Max(NA-test) <776.5 and Skewness(TCC-train) <0.317803 and Mean(Co-train) >=0.023858 and Skewness(NA-train) then Result= No	1118
if Max(NA-test) >=776.5 and Max(NOP-train) <1.04167 and std(OMMEC-train) >=53.4356 and Skewness(NIHICP-train) >=5.94918 and Max(CLD-train) >=77 then Result= No	460

RQ 5: Which pairs of source code metrics set and project data sets perform better to train the change-proneness prediction model for a project?

From Tables 5 and 6, we observe that the cross-project change-proneness prediction works in few cases, the %GP is very low. Hence, it is necessary to select best training data to train the models. The training data (source project) contains the data of other project consisting of software metrics and change-proneness information. Hence, it is necessary to select best set of source code metrics and project for cross-project change-proneness prediction. In this work, we apply genetic algorithm (GA) to select best training data consists of best set of metrics and projects. We apply the following sequence of steps to find the best training data using GA:

1. **Initialization of Chromosome**: In this study, the size of population is fixed i.e., number of chromosome (NOC) =100. Each chromosome consists of two parts. The first part of chromosome contain the source code metrics information. The second part of chromosome contain the cross project information. Each chromosome has binary encoded form contain only 0 and 1. The length of the chromosome L is calculated by applying the following formula:

$$L = (NOSM + NOP) \quad (1)$$

where $NOSM$ and NOP are the number of source code metrics and number of project used as cross-project prediction. In this paper, for each project the number of source code metrics is 61 and number of cross project is 9. Hence, the length of each chromosome is 70.

2. **Computation of Fitness Value**: In this work, our objective is to select the best set of source code metrics and project and enhance the overall performance of cross-project

prediction models. The fitness value (F) of each chromosome (selection of chromosome happens based on the fitness value) is calculated using the following formula:

$$F = 0.5 * acc + 0.25 * \frac{NOSSM}{NOSM} + 0.25 * \frac{NOSP}{NOP} \quad (2)$$

Where acc represents the accuracy of the developed model and NOSSM and NOSP represents the number of selected source code metrics and number of selected projects respectively. Our objective is to maximize the fitness value.

3. **Ranking the Chromosomes**: Based on the fitness value of chromosome, we rank the chromosome. High fitness value chromosome has rank 1.

4. **Cross Over and Mutation**: The probability of crossover is represented by Pc and the probability of mutation represented by Pm (which are basic operators of genetic algorithm). These values are kept constant i.e., Pc=0.5, and Pm=0.2.

5. **Stopping Criteria**: The execution of the algorithm terminates when last 20 generation best fitness value is same or number of generation reaches 200, beyond this level the fitness value of chromosome get almost saturated.

Figure 7 shows the selected set of source code metrics using GA for each projects. The horizontal and vertical line shows the name of project and source code metrics respectively. The filled circle shows the selected set of metrics using GA for a cross project prediction. From Figure 7, it can be seen that, NA, NumPara, SLOC, stms, AID, NMA, SP, SPA, SPD, CAMC, DCi, LCOM4, NHD, PCC, SNHD, TCC, AMMIC, CBI, CBO, DAC, DCAEC, IHICP, MPC, OCAEC, OMMEC, OMMIC, and RFC source code metrics are selected for compare dataset.

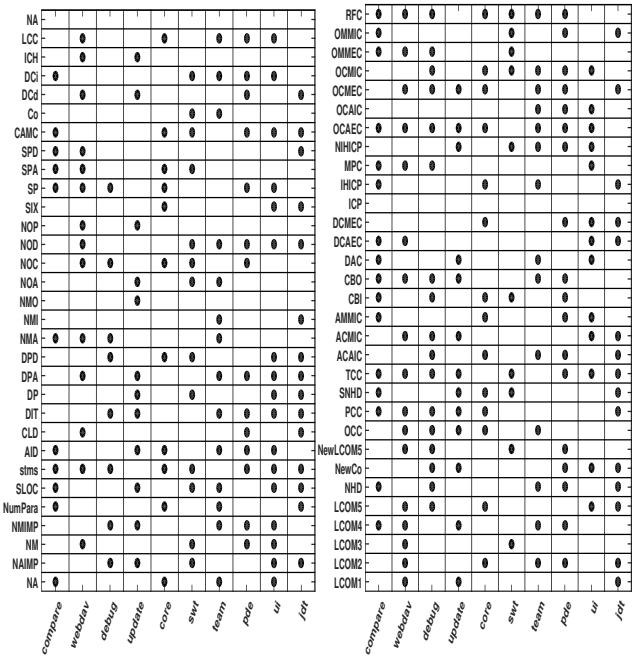


Figure 7: Selected Source Code Metrics using GA

Figure 8 shows the selected set of project data sets used to train the models. The horizontal and vertical line shows the name of project and selected projects respectively. The filled circle shows the selected set of project using GA for a cross project prediction. From Figure 8, it can be seen that, <webdav, pde> data set used to train the model for compare data.

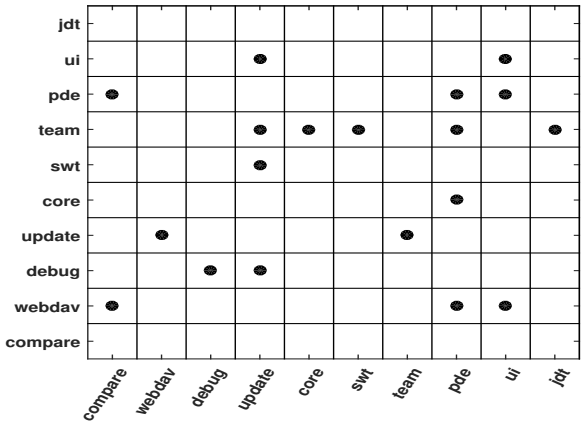


Figure 8: Selected Project using GA

Figure 9 shows the box-plot diagrams for different performance parameters of developed change-proneness prediction models using selected set of metrics and projects using GA. From Figure 9, it can be observed that the model developed using random forest (RF) yields better result as compared to other classifiers.

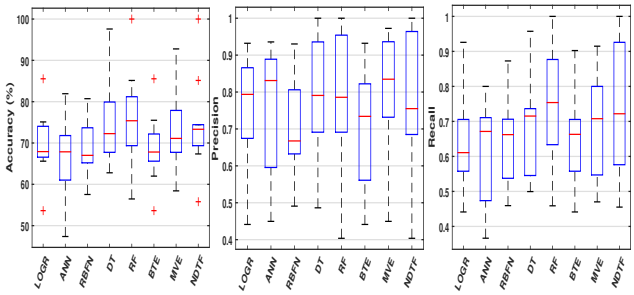


Figure 9: Performance of Change Proneness Model for Different Classifiers Trained using Selected Set of Metrics and Project using GA

Here, we have also considered Wilcoxon test with Bonferroni correction for comparison analysis. In this experiment, same eight different classifiers are used to developed change-proneness model. Each classifier is applied on ten different datasets. Therefore, for each technique a total number of four sets are employed, each with 10 data points (10 datasets * 1 best set of metrics and project using GA). The results of the pair-wise comparisons of different classifiers for accuracy performance parameter are shown in Table 8. The results for other parameters are of similar types. In this work, eight classifiers have been considered for analysis, i.e. total number of twenty eight (28) different pairs are possible (${}^{8technique}C_2 = 8 * 7/2 = 28$) and all results are analyzed at a 0.05 significance level. Hence, the null hypothesis "there is no significant difference between the two classifiers" is rejected only if $P - value \leq \frac{0.05}{28} = 0.0018$. From Table 8, it can be seen that, out of 28 pairs of training methods, 7 are found to have significant results. From Table 8, it is also observed that the RF yields better results compared to other classifiers based on the value of performance mean difference.

Table 8: Wilcoxon Test Results with Bonferroni Correction

p-value								
	LOGR	**ANN**	**RBFN**	**DT**	**RF**	**BTE**	**MVE**	**NDTF**
LOGR	1.00	0.32	0.28	0.02	0.00	0.63	0.03	0.00
ANN	0.32	1.00	0.70	0.00	0.00	0.74	0.00	0.01
RBFN	0.28	0.70	1.00	0.01	0.01	0.84	0.03	0.01
DT	0.02	0.00	0.01	1.00	0.19	0.01	0.73	0.77
RF	0.00	0.00	0.01	0.19	1.00	0.00	0.04	0.13
BTE	0.63	0.74	0.84	0.01	0.00	1.00	0.03	0.00
MVE	0.03	0.00	0.03	0.73	0.04	0.03	1.00	0.73
NDTF	0.00	0.01	0.01	0.77	0.13	0.00	0.73	1.00
Mean Difference								
LOGR	0.00	3.40	1.57	-4.58	-6.78	0.94	-4.14	-4.85
ANN	-3.40	0.00	-1.83	-7.98	-10.18	-2.46	-7.53	-8.24
RBFN	-1.57	1.83	0.00	-6.15	-8.35	-0.63	-5.71	-6.42
DT	4.58	7.98	6.15	0.00	-2.20	5.52	0.45	-0.26
RF	6.78	10.18	8.35	2.20	0.00	7.72	2.64	1.93
BTE	-0.94	2.46	0.63	-5.52	-7.72	0.00	-5.07	-5.78
MVE	4.14	7.53	5.71	-0.45	-2.64	5.07	0.00	-0.71
NDTF	4.85	8.24	6.42	0.26	-1.93	5.78	0.71	0.00

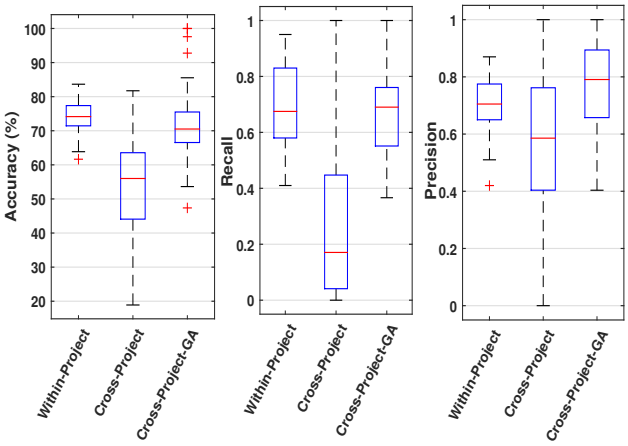


Figure 10: Within Project and Cross Project Change-proneness Prediction Model Performance

RQ 6: Is the performance of change-proneness prediction models developed using within project comparable to cross-project validation and is it statistically different than cross-project project change-proneness prediction?

In this work, finally we compared the performance of model developed using within project validation, cross-project validation and cross-project with genetic algorithm (cross-project-GA) validation method. Figure 10 shows the comparative results using different validation methods. From Figure 10, it can be observed that the within project validation method were found better than cross-project (without GA) validation method but not cross-project-GA. It can be further observed from the Figure 10 that the results of the cross-project-GA were found comparable or even better (precision and recall) than within project results.

Moreover, we also considered Wilcoxon test with Bonferroni correction to validate whether there is any statistical difference amongst the results obtained using with within project validation, cross-project validation and cross-project-GA validation. The results of the pair-wise comparisons of different combinations for accuracy performance parameter are shown in Table 9. The results for other parameters are of similar types. From Table 9, we observe that, the models developed using within project and cross-project are significantly different due to fat that p-value smaller then 0.0167. From Table 9, it is also observed that the within project validation method were found better than cross-project validation method based on mean difference. Finally, it is obsessed that the performance of cross-project-GA were found comparable result than within project results.

Table 9: Wilcoxon Test with Bonferroni Correction: Within Project and Cross Project Change-Proneness Prediction

	p-value			Mean Difference		
	Within-project	*Cross-project*	*Cross-project-GA*	*Within-project*	*Cross-project*	*Cross-project-GA*
Within-project	1.000	0.000	0.003	0.00	20.44	-2.99
Cross-project	0.000	1.000	0.000	-20.44	0.00	-17.45
Cross-project-GA	0.003	0.000	1.000	2.99	17.45	0.00

6. CONCLUSION

We conclude that for with-in project experimental setting, Random Forest (RF) technique results in the best precision. The Majority Voting Ensemble (MVE) performs better in comparison to BTE and NDTF. Our study reveals that the accuracy differs across the eight classifiers for within-project setting. Our analysis shows that in general the within-project accuracy results are better than the cross-project accuracy results. In case of cross-project change-proneness prediction, our analysis reveals that the NDTF ensemble method performs higher than the other individual classifiers and ensemble methods in the experimental dataset. We observe that the ratios of successful cross-project change-proneness prediction ranges from 0.02% to 15.53% based on acceptable prediction result criteria in literature (recall ≥ 0.7 and precision ≥ 0.5). Our analysis conducted based on decision tree induction on the distributional characteristics of the source code metrics shows that the characteristics of source code metrics is a valuable indicator and can be used to take decision regarding selection of suitable training data for cross-project change-proneness prediction. We apply genetic algorithm (GA) to select best training data consists of best set of metrics and projects. Application of GA results in selected set of metrics and project resulting in better performance. We conduct a comparison of within-project, cross-project without GA and cross-project with GA and our analysis reveals that cross-project with GA performs best followed by within-project and then cross-project without GA.

7. REFERENCES

[1] Canfora, G., De Lucia, A., Di Penta, M., Oliveto, R., Panichella, A., and Panichella, S. Multi-objective cross-project defect prediction. In *Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on* (2013), IEEE, pp. 252–261.

[2] Chen, L., Fang, B., Shang, Z., and Tang, Y. Negative samples reduction in cross-company software defects prediction. *Information and Software Technology 62* (2015), 67–77.

[3] Demšar, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research 7*, Jan (2006), 1–30.

[4] Eski, S., and Buzluca, F. An empirical study on object-oriented metrics and software evolution in order to reduce testing costs by predicting change-prone classes. In *Software Testing, Verification and Validation Workshops (ICSTW), 2011 IEEE Fourth International Conference on* (2011), IEEE, pp. 566–571.

[5] Fukushima, T., Kamei, Y., McIntosh, S., Yamashita, K., and Ubayashi, N. An empirical study of just-in-time

defect prediction using cross-project models. In *Proceedings of the 11th Working Conference on Mining Software Repositories* (2014), ACM, pp. 172–181.

[6] He, Z., Shu, F., Yang, Y., Li, M., and Wang, Q. An investigation on the feasibility of cross-project defect prediction. *Automated Software Engineering 19*, 2 (2012), 167–199.

[7] Herbold, S. Training data selection for cross-project defect prediction. In *Proceedings of the 9th International Conference on Predictive Models in Software Engineering* (2013), ACM, p. 6.

[8] Kumar, L., Jetley, R., and Sureka, A. Source code metrics for programmable logic controller (plc) ladder diagram (ld) visual programming language. In *Proceedings of the 7th International Workshop on Emerging Trends in Software Metrics* (2016), ACM, pp. 15–21.

[9] Kumar, L., Rath, S. K., and Sureka, A. Empirical analysis on effectiveness of source code metrics for predicting change-proneness. In *Proceedings of the 10th Innovations in Software Engineering Conference* (2017), ACM, pp. 4–14.

[10] Kumar, L., Rath, S. K., and Sureka, A. Using source code metrics to predict change-prone web services: A case-study on ebay services. In *Machine Learning Techniques for Software Quality Evaluation (MaLTeSQuE), IEEE Workshop on* (2017), IEEE, pp. 1–7.

[11] Kumar, L., and Sureka, A. Using structured text source code metrics and artificial neural networks to predict change proneness at code tab and program organization level. In *Proceedings of the 10th Innovations in Software Engineering Conference* (2017), ACM, pp. 172–180.

[12] Ma, Y., Luo, G., Zeng, X., and Chen, A. Transfer learning for cross-company software defect prediction. *Information and Software Technology 54*, 3 (2012), 248–256.

[13] Malhotra, R., and Bansal, A. J. Cross project change prediction using open source projects. In *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on* (2014), IEEE, pp. 201–207.

[14] Malhotra, R., and Khanna, M. An empirical study for software change prediction using imbalanced data. *Empirical Software Engineering* (2017), 1–46.

[15] Yan, M., Fu, Y., Zhang, X., Yang, D., Xu, L., and Kymer, J. D. Automatically classifying software changes via discriminative topic model: Supporting multi-category and cross-project. *Journal of Systems and Software 113* (2016), 296–308.

[16] Zhu, X., Song, Q., and Sun, Z. Automated identification of change-prone classes in open source software projects. *JSW 8*, 2 (2013), 361–366.

[17] Zimmermann, T., Nagappan, N., Gall, H., Giger, E., and Murphy, B. Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering* (2009), ACM, pp. 91–100.