Heterogenous Defect Prediction using Ensemble Learning Technique

Arsalan Ahmed Ansari
Department of Computer Science and Engineering
National Institute of Technology
Rourkela, India
217CS3434@nitrkl.ac.in

Amaan Iqbal
Department of Computer Science and Engineering
National Institute of Technology
Rourkela, India
116CS0186@nitrkl.ac.in

Dr. B. D. Sahoo
Department of Computer Science and Engineering
National Institute of Technology
Rourkela, India
bdsahu@nitrkl.ac.in

ABSTRACT

Software Defect Prediction (SDP) is the most practically used approach in the testing phase of the software development life cycle (SDLC) which helps to find out the defected module prior to testing or releasing the product. This study intends to predict the defects through an improved Heterogeneous Defect Prediction approach based on Ensemble Learning Technique which consists of 11 different classifiers. This study analyses the way Ensemble Learning technique which includes the combination of both supervised and unsupervised machine learning algorithm helps in predicting the defect proneness of modules. This technique has been applied on historical metrics dataset of various projects of NASA, AEEEM, and ReLink. The data set is sourced from the PROMISE repository. The performance of the obtained models is critically assessed using the Area under the Curve, precision, recall, f-measure. Experiment results shows that our method is comparable to the existing method for defect prediction.

Keywords— Software Defect Prediction (SDP) ; Within Project Defect Prediction (WPDP); Cross Project Defect Prediction (CPDP); Heterogeneous Defect Prediction (HDP); Ensemble Learning.

I. INTRODUCTION

Software Defect Prediction is an approach to find out the defected modules of a software much earlier than the testing phase of SDLC, so that the testing resources can be utilized efficiently. There are three major categories of SDP: Within Project Defect Prediction (WPDP), Cross Project Defect Prediction (CPDP), Heterogeneous Defect Prediction(HDP).

In WPDP, the model is trained with the labelled instances of a project and predict new instances of the same project. As the industry is evolving with a rapid pace, new types of projects are building, in those cases the labelled instances of the same projects are not available for the training of a model. So a developer seeks some expertise from his or someone else's past experiences and tries to predict the defect proneness of modules of new project. This concept is generally called as Transfer Learning (TL). Based on TL, researchers have proposed CPDP, in which model predicts defect proneness for new projects lacking in historical data by taking the advantage of expertise available from other projects. The only limitation of CPDP is that it only works with same metrics set between projects. It is a challenging task to collect same metrics data due to the frequent occurrence of paradigm shifts in the software industry.

HDP [14] is an approach to predict defects across projects even with heterogeneous metric sets (No common metrics/features between source and target projects). HDP is the one of the most recent research area in the field of Software Engineering.

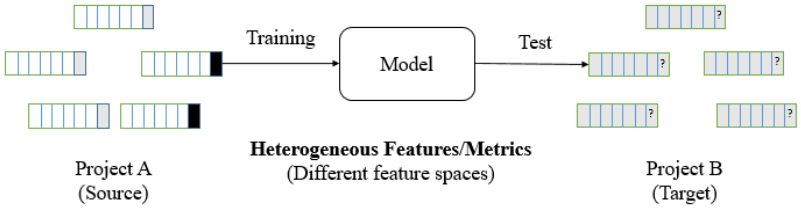

Figure 1: Illustration of HDP

Fig. 1 shows that there are two different projects having heterogeneous metrics set. Each blocks consists of many blocks which contains the metrics / features values. The last column represents module label as defective or non-defective. Non-defective is represented using grey color box while the defective is represented using black. The question mark represents the unknown labels. As shows in the figure the module is trained using the labelled instances of a project and predicts defects on unlabeled instances of other projects having heterogeneous metrics set.

In this paper, three different statistical methods for metrics matching along with the Ensemble Learning Classifier (consists of 11 different classifier) have been implemented and their performances are assessed with the previous existing approach.

II. LITERATURE SURVEY

Software defect prediction using machine learning has been in the foray since 1985 when Shen et al. [1] proposed first prediction model using regression, although at that time the defect prediction was much easier because the Cross Project and Heterogeneous Defect predictions were not present earlier. Since then the researchers have been continuously researching in this area. Munson et al. [2] claimed that regression techniques were inefficient so they suggested a new classification model which divided the components into two different sections: high risk and low risk and achieved an accuracy of 92% on their subjective system.

Lessmann et al. [3] conducted an experiment to benchmark the 22 classification models for software defect prediction. They concluded that, not a single technique outshines the other.

Machine Learning heavily depends on the historical data, what if the previous data of the task undertaken is not available. To resolve this issue, researchers proposed a new technique called Cross Project Defect Prediction which exploits the historical data of one or multiple already existing projects and predicts defect of new project which is lacking in historical data [4] [5] [6].

Sheppard et al. [7] conducted an analysis of six hundred experimental outcomes taken from 42 initial findings and concluded that there is no uniformity amongst the researchers in reporting the performance of classification One of the interesting thing they found that only 1% variability in the performance of prediction system would be credited to the algorithms the rest 99% is for the researcher.

Venkata U.B. Challagulla et al. [8] did an experimental analysis of machine learning based software defect prediction techniques but they did not consider cross project and heterogeneous defect prediction. They applied various machine learning techniques such as Linear Regression, Pace Regression, Support Vector Regression, Neural Network for continuous goal field, Support Vector Logistic Regression, Neural Network, Logistic Regression, Naive Bayes, Instance Based Learning, K-nearest neighbors, J48 Trees, and 1-Rule. Amongst all the techniques, Instance based learning and 1-Rule outshines the others.

Challagulla et al. [6] also noticed that classifying the modules as defective or non-defective is a step ahead than forecasting the actual error in dataset. They also noticed that there was an irregularity in the rank of learning techniques in terms of forecast accuracy across different data sets which means it all depends on the datasets. Some techniques performed for some dataset(s) but not for all. But they tried to give best possible techniques and from their analysis they concluded that Naive Bayes, Instance based learning(IBL), and neural networks are the improved prediction models as compared to the other models. Almost all the machine learning techniques have been applied by the researchers for software defect prediction. Some of them performed good for some dataset some of them are unstable. Gan et al. [9] noticed that ability to forecast based on SVM is precarious so they proposed a new technique where they are using Grey Relational Analysis before Support Vector Machine (GRA-SVM) which improves the overall accuracy of the software defect prediction.

Cross project defect prediction works only when the metrics/features is identical between two projects. The evolution in the industry is leading to very brand new software systems which sometimes leads to heterogeneous metrics dataset for defect prediction algorithm.

III. METHODOLOGY ADOPTED

A. Dataset Considered

In this paper, four groups of datasets are used: NASA, AEEEM, ReLink, and SOFTLAB. These datasets are standard dataset that have been used by many researchers for the defect prediction studies over time.

Table 1: Four groups of dataset with instance and metrics detail

Organization	Dataset	Number of instances		Number of metrics
		All	Defective	
NASA [10]	JM1	9593	1759	21
	MC2	127	44	39
	KC3	200	36	39
AEEEM [11]	EQ	324	129	61
	JDT	997	206	
	ML	1862	245	
ReLink [12]	Apache	194	98	26
	Safe	56	22	
	ZXing	399	118	
SOFTLAB [13]	ar4	107	20	29
	ar5	36	8	
	ar6	101	15	

The table 3 shows the details about the dataset. It shows that there are four groups which contains 3 datasets each, having combination of both defective and non-defective instances. For ex. the dataset JM1 is provided by NASA which contains the total of 9593 instances out of which 1759 are labelled as defective instances and it has 21 metrics/features.

Table 2: Common metrics between of group of dataset

Dataset pair		Number of common metrics
NASA	SOFTLAB	28
NASA	ReLink	3
NASA	AEEEM	0
SOFTLAB	ReLink	3
SOFTLAB	AEEEM	0
ReLink	AEEEM	0

Table 2 shows the common metrics between a group of dataset. Half of the pairs of datasets do not have a single common metrics, for example, NASA dataset has 3 common metrics with ReLink dataset. Those common metrics is removed from both the groups of dataset so that the dataset becomes heterogeneous in nature.

B. Feature Selection

There are various feature selection techniques like chi-square, recursive feature elimination, recursive feature elimination - cross validation, gain ratio, f-classif, significance attribute evaluation, mutual info classif. After conducting an experiment on each technique for selecting the most significant metrics, chi-square outshines the other. The top 30% metrics is selected from both source and target dataset. Chi-square test has been applied to find the dependency between features and its labels on both source and target dataset and then selected top 30% of metrics. Since the metrics in HDP dataset is heterogeneous, so the selected metrics from both the dataset doesn't have a single common feature between them.

C. Metric Matching

Metric Matching is the core part of Heterogeneous Defect Prediction. It solves the problem of heterogeneity in Cross Project Defect Prediction. The basic idea behind metric matching is to find the best similar metrics which have higher values and have higher tendency for a module being defective. There are various techniques available for metric matching. Jaechang Nam et al. conducted an experiment on various techniques and concluded that Kolmogorov Smirnov Test based matching technique performed better than the other.

Kolmogorov Smirnov Test: It is a non-parametric test of equality which can be used to compare two samples based on the correlation between them. To find the best matched metrics multiple pairs of samples is taken and after applying KS-test, p-value (probability value) from the KS-test statistics have been taken to find the best matched metrics. p-value interpretation is similar to another hypothesis test, which is generated with the help of the test statistics of KS-test which is calculated by:

$$D_{n,m} = sup\left|F_{1,n}(x) - F_{2,m}(x)\right| \quad (1)$$

Where $F_{1,n}$ and $F_{2,m}$ are the distributions of two samples and n and m are the size of both the samples undertaken and sup is supremum function.

With the help of test statistic, the p values are looked up from the KS-test p-value table.

D. Maximum Weighted Bipartite Matching

Result from KS- test is in the form of metrics where each values represents the p-value for every pair of metrics. Maximum Weighted Bipartite Matching is a way to find the best matched metrics from the p-value matrix. We have used Ford Fulkerson algorithm which uses the Flow Network to find the maximum flow. The idea to find the metrics pair came from the very famous problem of finding the best applicant for a job which was solved using the above said algorithm.

E. Feature Transformation

Before building the prediction model, feature transformation technique can be used to reduce some features by combining possibly correlated features to save the computation cost. After conducting an experiment between Linear Discriminant Analysis (LDA) and Principal Component Analysis (PCA), it was found that LDA didn't prove to be helpful maybe because it is applicable to linear datasets i.e., data that can be distinguished by linear decision boundary. In this paper, PCA is used for the feature transformation.

F. Ensemble Learning Technique

In Machine Learning, sometimes different classifiers give different results for a single instance based on their learning capability. Ensemble Learning is a method to solve that problem. It is one of the most powerful way to boost the performance and accuracy of a model. Voting classifier is one of the way of Ensemble Learning where votes of different classifier is taken and on the basis of votes, class label is decided for that instance. In this approach, 11 different classifiers are used to decide the majority, they are as follows:

- Logistic Regression
- Support Vector Machine (linear kernel)
- Support Vector Machine (RBF kernel)
- K Nearest Neighbors
- Naïve Baye's
- Decision Tree
- Random Forest Classifier
- Ada-Boost
- XG-Boost
- Multi-layer Perceptron
- Probabilistic Neural Network

IV. IMPLEMENTATION

Proposing an ensemble learning algorithm, the aim is to ensure a robust method to ameliorate the caliber of the software product. The implementation details of the algorithm using flowchart are discussed below:

A. Pre-processing the dataset and Model Training

For training the prediction model, the number of metrics between source and target dataset must match, but in HDP dataset, number of metrics available is not uniform. To make it uniform and to select the most significant metrics, feature selection technique is applied.

After feature selection, feature scaling is applied to normalize the dataset. Feature scaling is one out of many essential pre-processing steps if the metric values in dataset are not in the standard range. Like in the dataset which we are using, if we look at the Cyclomatic complexity metric and LOC, they differ greatly in range.

Now KS-test is applied to the top 30% metrics from both the source and target dataset to find the best matched metrics with the help of Maximum Weighted Bipartite Matching. After applying, the above techniques,

the dataset for training and testing is ready. The Ensemble Voting Classifier is trained with the source dataset and tested on the target dataset.

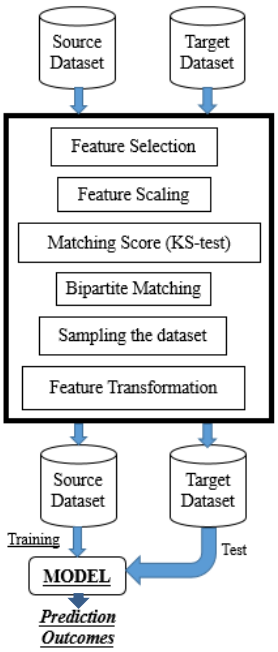

Figure 2: HDP approach overview

B. **Evaluation Critera**

The defect prediction models are assessed by the values of the area under the curve (AUC) to compare with the previous existing technique. Models are also judged by mean squared error (MSE), precision, recall, f1-score to show the errors in the prediction models. The formula for the calculation of the above measures are given as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Actual_i - Predicted_i)^2 \quad (2)$$

To understand below performance measure, first understand various prediction outcomes:

- True Positive (TP): buggy instances predicted as buggy
- False Positive (FP): clean instances predicted as buggy
- True Negative (TN): clean instances predicted as clean
- False Negative (FN): buggy instances predicted as clean

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

$$Recall = \frac{TP}{TP+FN} \quad (4)$$

$$F1 - score = \frac{2*(Precision*Recall)}{Precision+Recall} \quad (5)$$

V. RESULT AND DISCUSSION

Around 67 combinations of source and target dataset were made and whose AUC value along with other performance measures are higher are displayed in the table no. 3. The table rows consist of the source dataset by which the prediction model is trained and the target dataset on which the predicted model is tested along with values of performance measures like AUC, MSE, Precision, Recall, and F1-Score.

Table 3: Prediction result of source and targets in terms of best AUC values

Source	Target	AUC	Mean Squared Error	Precision	Recall	F1-Score
KC3	EQ	0.79	0.29	0.71	0.71	0.71
JM1	JDT	0.80	0.19	0.82	0.81	0.81
JM1	ML	0.74	0.31	0.84	0.68	0.73
EQ	Apache	0.71	0.29	0.71	0.71	0.71
EQ	Safe	0.80	0.23	0.77	0.77	0.77
JDT	ZXing	0.67	0.32	0.67	0.68	0.67
ML	JM1	0.73	0.27	0.76	0.73	0.74
ML	MC2	0.70	0.29	0.69	0.70	0.69
EQ	KC3	0.74	0.39	0.91	0.61	0.69
EQ	AR4	0.83	0.36	0.86	0.64	0.67
ML	AR5	0.93	0.13	0.89	0.86	0.87
EQ	AR6	0.69	0.44	0.85	0.55	0.61

In our study as shown in the table, it is found that EQ dataset from AEEEM group as a source dataset works best with almost half of the dataset. The results shown in the table is 1% - 15% higher than existing method for HDP.

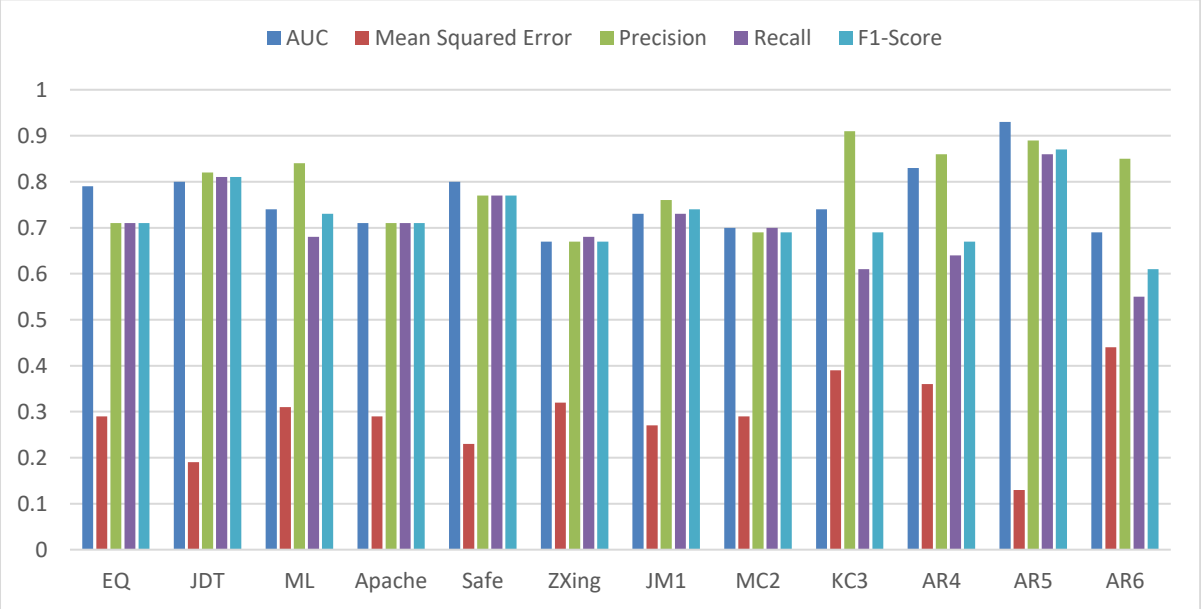

Figure 3: Graphical representation of Table 3

Figure 3 represents the bar chart representation of the table 3. The representation for a dataset is grouped in terms of the target dataset. For ex. the AUC, MSE, Precision, Recall, F1-Score of EQ is shown at one place.

The datasets are highly complex in nature, it has the highest degree of non-linearity and the datasets also has the class imbalance problem. Heterogeneous Defect Prediction after applying various pre-processing techniques with ensemble voting classifier gives promising results.

VI. CONCLUSION AND FUTURE WORK

By using various classifier in an ensemble voting technique for heterogeneous defect prediction with effective preprocessing techniques gives the promising result. which will encourage the researchers to use this technique in various machine learning problems.

In this era, it has become important to sort and manage the test analysis work as it enhances the possibilities of the forecasting the errors as early as possible which in turn benefits the organization as well as helps in building liaisons with the customers. The simple aim of identifying the faults to lessen the tedious work of the developers, propels the researchers into the onus of ensuring that all the faults are identified and sorted out by the programmers within the tight deadline put by the organization. In this paper, we have put an effort in analyzing the ensemble learning technique which provides us with the outcomes which are comparable with the baseline method but it the change in performance is not drastic. To improve the accuracy of the prediction model, our future work will be based on some deep learning techniques for metric matching and we will be trying some other feature selection techniques also.

REFERENCES

[1] V. Y. Shen, T. J. Yu, S. M. Thebaut, and L. R. Paulsen, "Identifying error-prone software—an empirical study" IEEE Transactions on Software Engineering, vol. 4, pp. 317--324, 1985.

[2] J. C. Munson, and T. M. Khoshgoftaar, "The detection of fault-prone programs" IEEE Transactions on Software Engineering, vol. 18, no. 5, pp. 423--433, 1992.

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

[4] Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross-company software defect prediction" Information and Software Technology (Elsevier), vol. 54, no. 3, pp. 248--256, 2012.

[5] J. Nam, S. J. Pan, and S. Kim, "Transfer Defect Learning" 35th IEEE International Conference on Software Engineering (ICSE), pp. 382-391, 2013

[6] B. Turhan, T. Menzies, A. B. Bener, and J. D. 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, 2009. Springer

[7] M. Shepperd, D. Bowes, and T. Hall "Researcher bias: The use of machine learning in software defect prediction" IEEE Transactions on Software Engineering, vol. 40, no. 6, pp. 603--616, 2014.

[8] V. Challagulla, F.B. Bastani, and I. L. Yen "Empirical assessment of machine learning based software defect prediction techniques" International Journal on Artificial Intelligence Tools, World Scientific. Vol. 17, no. 2, pp. 389--400, 2008.

[9] Y. Gan, and C. Zhang "Research of software defect prediction based on GRA-SVM" AIP Conference Proceedings, Vol. 1890, no. 1, pp. 040116, 2017.

[10] T. Menzies, et al., "The promise repository of empirical software engineering data," Jun. 2012. [Online]. Available: http:// promisedata.googlecode.com

[11] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: A benchmark and an extensive comparison," Empirical Softw. Eng., vol. 17, no. 4/5, pp. 531–577, 2012.

[12] R. Wu, H. Zhang, S. Kim, and S. Cheung, "ReLink: Recovering links between bugs and changes," in Proc. 16th ACM SIGSOFT Int. Symp. Found. Softw. Eng., 2011, pp. 15–25.

[13] B. Turhan, T. Menzies, A.B. Bener, andJ.Di Stefano, "On therelative value of cross-company and within-company data for defect prediction,"EmpiricalSoftw.Eng.,vol.14,pp.540–578,Oct.2009.

[14] J. Chang, W. Fu, S. Kim, T. Menzies, and L. Tan, "Heterogeneous Defect Prediction", IEEE Transaction on Software Engineering, vol. 44, no. 9, pp. 874-896, 2018