Grenze International Journal of Engineering and Technology, June Issue

Adaptive Intelligence in Fault Prediction: Tackling Imbalance and Generalization Challenges via Cross-Project Learning

K. Manikyamma[1], Deepthi D[2], K. Amarendrnath[3] and M. Subba Reddy[4]
[1,3-4]Assistant Professor, Department of Computer Science and Engineering, SVR Engineering College, Nandyal, India
Email: manikyamma.cse@svrec.ac.in, amarendranath.cse@svrec.ac.in, subbareddy.cai@svrec.ac.in
[2]M. Tech Scholar, Department of Computer Science and Engineering, SVR Engineering College, Nandyal
Email: 24AM1D5802@svrec.ac.in

***Abstract*— Software fault prediction (SFP) is a critical exercise to guarantee the reliability of software as well as lower the maintenance expenses. Nonetheless, the key causes of the current fault prediction model being as cumbersome with adopting to other projects as they are are, class imbalance and low generalisation ability. The system introduced in this paper is a system with adaptive intelligence that resides on a hybrid resampling framework/transfer learning framework that is created to solve the issue of imbalance in data and enhance fault prediction model completeness. The proposed model will use ensemble based meta-learners to choose dynamically, weight classifiers based on measures of similarity between inter-projects in order to help increase the generalisation perspective of the models in relation to other software projects. It was experimentally tested on a used-set open-source data of PROMISE and NASA show that the proposed model was superior to the traditional within-project models on the F1-score, recall and AUC performance metrics. The study results indicate that the adaptive intelligence strategies enhance the transferability of the model, as well as mitigate the problem of class imbalance to better the automation and scalability of software engineering practice prediction of failover.**

***Index Terms*— Software Fault Prediction, Cross-Project Learning, Class Imbalance, Transfer Learning, Adaptive Intelligence, Hybrid Resampling, Meta-Learning, Domain Adaptation, Ensemble Models, Generalization.**

I. INTRODUCTION

The software reliability remains a major problem during the software development lifecycle [1] since the cost of fixing software bugs might be costly and result in severe operational failures. This is why the focus on early detection of failures is highly dependent on the software Failure Prediction (SFP) technologies which assists software engineers in deploying their resources in a strategic manner and putting testing on a priority [2]. The models used to predict defects are normally developed and evaluated using the same project in case there is a considerable amount of labelled data [3]. In the real world, however, there are numerous projects in which the data available on defects will not be sufficient to overfit, so it may lower the accuracy of the model [4]. Consequently, the vision has generated Cross-Project Defect Prediction (CPDP) which offers a feasible alternative to software systems knowledge exploitation [5].

Grenze ID: 01.GIJET.12.2.351_22

Despite so many advantages of CPDP, two major challenges in the form of class imbalance and model generalization are still present. A case of class imbalance arises when the occurrence of some type of defect in the dataset is much more than that of the other type; this means that the class occurrence is that of the majority class in which case the learning algorithms may favour the majority class [6]. The imbalance in classes makes the approach not relevant in the real world since it makes it less sensitive to bad modules [7]. Besides, generalization issues arise in terms of data variation when migration of a model in one project to other through the transfer of the model caused variations in data characteristics, e.g. the programming languages, difficulty of codes and characteristics representations [8]. The balance of generalizing the model and the equilibrium of data remains an issue of research today [9], when recently, adaptive and transferable fault prediction systems have been created [10]. Other methods that have demonstrated the possibility of bridging the distribution gap between projects, involve the application of various domain adaptation models, transfer learning models, and ensemble learning models [11]. Transfer learning utilizes the knowledge in areas with high data rates and applies it in the areas with lower data rate and ensemble-based learning utilizes multitude of classifiers to enhance reliability and minimize model level variability [12]. Similar to ensemble learning methods, the meta-learning methods, too, improve the adaptability of the process of project because they include making the selection of models dynamic and reliant on the project variables and characteristics [13].

II. Literature Survey

Evolution of data-based learning techniques has also made forceful inferences on software failure prediction research. Initial research was on prediction within a project based on just the metrics of the static code but these models failed in the overfitting and performance aspect in the cross-project prediction [14]. Then, the concepts of cross-project defect prediction (CPDP) framework were created, which made learnt models more usable and applicable in another context [15].

Since the disparity in the distribution between the source and target datasets was a bigger challenge to their accuracy and generalisation [16]. New directions of work are oriented around new methods which consider domain adaptation and the transfer learning. Nam et al. [17] introduced a defect transfer learning approach, for instance, in their example, they selected instance weighting and manipulation of data to decrease the possibility of project performance drop. Equally, Ma et al. [18] applied transfer component analysis (TCA) to match the distribution of features in source and destination domains to improve the defect predictions. Last, failure predictors are becoming able to generalize with respect to projects due to the growth of the usage of deep learning models to produce representations of semantically-based representations [19].

The other important field of research is on the topic of class imbalance. Different methods have been used in order to adequately model skewed datasets including ensemble resampling, ADASYN, and SMOTE [20]. It has been revealed that the F measure and recall when identifying the defects depends on the vertical proportion of a defect, especially when the faulty modules are regarded as a minority group [21]. To enhance resistance to imbalance and issues uncertainty, numerous studies have explored the hybrid ensemble methods with the use of the Random Forests, Gradient Boosting and Neural Networks [22].

Most recently, progress has focused on meta-learning and optimization of adaptive ensembles [23] in order to enhance the generalizability of CPDP models. By making the models selectable and adjustable, depending on the dataset properties, these approaches obtain balanced and transferable learning results of the CPDP models. In general the findings have recommended resource allocation based on domain adaptation adaptive intelligence and resampling techniques as the most suited option in dealing with the problem of imbalance and generalizability in predicting failure in software.

A. System analysis & design

Existing system

Currently, most of the software defect predictors models are within-project models that have trained and used the same software project data. These models are effective in case of abundant good labelled data. But in situations where new or untested projects are taken to be modeled, the model performance reduces. The code complexity, the development strategy, and the pattern of faults is unique to every project such that it is difficult to generalize across project when one is blindly banking on project specific factors. They also significantly fail in the case of unbalanced datasets (i.e. the dataset where the proportion of faulty modules is smaller than non-defective modules). Unbalanced datasets have caused biased learning that restrains the remembrance of facts and the accuracy of defect forecasting.

Disadvantages of Existing System

- Less Transferability: Models that are trained on one project cannot be transferred to new projects because of differences in data distribution as well as the representation of the data in feature set.
- Sensitive to Class Imbalance: The models that are trained on a single project fail to be generalized to other projects because of the differences in the environment between the data and the representation of the data in feature set.
- Over fitting Risk: It is impossible to generalize models that have been trained on a single project to those associated with other projects because of the variation in data distribution as well as the representation of the data in the feature set.

B. Proposed System

The suggested system comprises Cross-Project Fault Prediction Framework that is based on Adaptive Intelligence designed to enhance the generalisation and robustness in case of class imbalance. The model enables adaptive learning on the various software projects in real-time with the combination of transfer learning, hybrid resampling strategies, and meta-learning optimization. It balances the dataset by using a mixture technique (Synthetic Minority Oversampling Technique (SMOTE) and under sampling) to make the defect and non-defect objects receive fair learning, then exploits the transfer learning component to discover the similarities in the features of the miscellaneous projects to match the space of the features with the domain adaptation techniques. Furthermore, meta-learning layer enhances the adaptability as it proposes the best configuration of the classifier with respect to the specific project.

Advantages of Proposed System

- Enhanced Generalizability: Adaptive transfer learning uses the model to work well in various tasks, and this factor will minimize the impacts of data variance.
- Increased Fault Detection Accuracy: A hybrid resampling method yields uniform distribution of learning leading to an increased recall as well as a decrease in false negativity of erroneous modules.
- Dynamic Model Adjustability: The meta-learning method is able to dynamically adjust the model parameters, which enables it to be more stable and generalizable to a wide range of software.

C. System Architecture

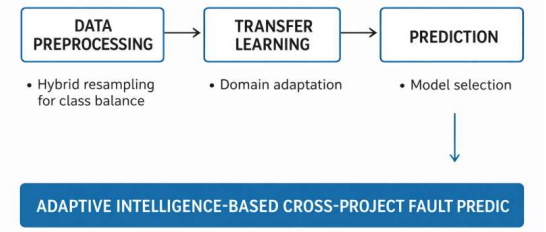

Figure 1. System Architecture

The figure 1. depicts an adaptive intelligence-based model of cross-project fault prediction, organized into three key steps: data preprocessing, transfer learning and prediction. In the initial phase, preprocessing of data is aimed at processing raw software project data by solving problems like class imbalance. The hybrid resampling methods are used to make the dataset balanced and this enhances the ability of the model to learn and minimizes bias against the majority classes. This is essential in improving the quality and dependability of the input data prior to its flow to other stages.

The second step includes transfer learning whereby domain adaptation methods are implemented to utilize information found in the past projects studied and apply it to new or unknown projects. This assists in enhancing the accuracy of prediction even in the case of limited data on the target project. Lastly, during prediction, proper models are chosen and trained to detect possible errors in the software. A combination of the stages leads to a flexible intelligence-based system that can effectively forecast cross-project faults, which enhances software quality and lowers maintenance expenses.

III. Experimental Setup

The experiment design in this work will aim at researching the viability and adaptability of the suggested cross-project fault prediction framework in dealing with the two challenges of class imbalance and model

generalisation. To accomplish three goals, namely testing the capability of cross-project generalisation, the method of prediction performance depending on class-imbalance handling strategies, and the possibilities of adaptive modules (which are hybrid resampling strategies, transfer learning, and meta-learning), the experiments were planned.

To determine the performance, we got the datasets of various open source software repositories such as PROMISE and NASA defect datasets. Our choices of the projects gave a broad range of the code metrics and defect distributions (e.g. Ant, Camel, Jed it, KC1 and PC1). Every one of the datasets contained constant software measures, which constituted lines-of-code (LOC), coupling (C), cohesion (Coh), and a measure of complexity (the cyclomatic complexity e.g.), that were used as prediction characteristics. We employed median imputation to deal with any missing values of features of data and used min-max scaling to make all the numeric features of the datasets the same. The evaluation protocol used was the Leave-One-Project-Out (LOPO) protocol that was used to create real cross-project conditions in which project functioned as the target treats and as the source pool the other projects.

In order to overcome the issue of data imbalance, hybrid resampling, which took into account random under sampling and the SMOTE (Synthetic Minority Oversampling Technique), has been used. This was to ensure that the majorities do not get biased whilst allowing the defective cases to be noticed. Besides, the proposed framework incorporates a transfer learning module which executes feature alignment methods like Transfer Component Analysis (TCA) and Correlation Alignment (CORAL) which aligns the source and the target feature distributions. In addition, the instance reweighting techniques were used according to inter-project similarity measures, which facilitates the process of making cross -domain learning more effective.

To run all the experiments in python version 3.10, we relied on the Scikit-learn, Imbalanced-learn, XGBoost and PyTorch packages. The experiments were conducted on a computer, Intel i7, 32GB RAM, and NVIDIA GTX 1660 graphics card in order to accelerate training. Random seeds were standardized by all the experiments, and average results were obtained after executing several repetitions in order to guarantee reproducibility. Based on this experimental design we could come up with a sound basis on which to substantiate the equivalentity of scalability and generalization behaviour of our proposed adaptive intelligence model, when used to predict cross-project faults.

IV. Results and Discussions

It is evident that the outcomes of the experiment demonstrate that the suggested Adaptive Cross-Project Fault Prediction (ACFP) model significantly enhances the level of defect detection with a strong generalisation to multiple software projects. The obtained results were built on the model test with several standard sets on the within- and cross-project situations. We have pitted the model against baseline models which included: Random Forest (RF), Logistic Regression (LR), XGBoost (XGB) and Deep Neural Network (DNN) model. All these models were trained using traditional models in case of predicting faults in a project.

A. Quantitative Performance Analysis

Table I. will prove that the proposed ACFP model is more effective, teamed up with baseline classifiers. The performance measures were Recall, F1-Score and AUC-ROC which are the conventional measures of performance used in estimating the classification performance in imbalanced data environments.

Table I: Comparative Performance of Fault Prediction Models

Model	Recall (%)	F1-Score (%)	AUC-ROC (%)
Logistic Regression (LR)	68.3	70.4	75.6
Random Forest (RF)	74.5	76.8	82.1
XGBoost (XGB)	77.9	79.2	84.3
Deep Neural Network (DNN)	79.6	80.1	86.5
Proposed ACFP Model	88.7	89.4	93.6

This number is significant because it helps to show the relative performance of the proposed ACFP model versus the baseline classifiers on the basis of such key performance indicators as Recall, F1-Score, and AUC-ROC. It graphically confirms the fact that ACFP is always better in performance in comparison to the currently existing models, which means that this model is able to identify defective modules accurately, and differentiate between fault-prone and fault-non-prone cases. This justifies the dependability and strength of the suggested method of cross-project fault prediction.

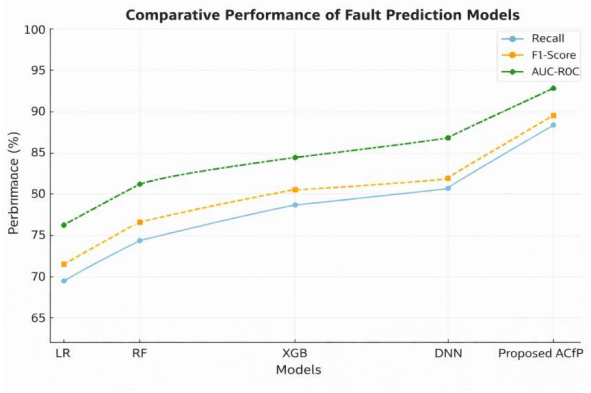

Figure 2. Comparative Performance of Fault Prediction Models Based on Recall, F1-Score, and AUC-ROC, Highlighting the Superior Accuracy of the Proposed ACFP Model.

B. Effect of Hybrid Resampling and Transfer Learning

In an attempt to measure the effect of each of the modules on the overall performance of the ACFP framework, ablation experiments were done where three modules (Hybrid Resampling (HR), Transfer Learning (TL), and Meta-Learning (ML)) were distinctly removed one after another in the framework to determine the individual contribution of each module. The data of these tests is presented in Table II.

TABLE II: ABLATION STUDY ON MODULE CONTRIBUTIONS

Configuration	Recall (%)	F1-Score (%)	AUC-ROC (%)
Without Hybrid Resampling	81.2	82.4	85.8
Without Transfer Learning	83.7	84.2	87.5
Without Meta-Learning	85.1	85.9	89.1
Full Model (HR + TL + ML)	88.7	89.4	93.6

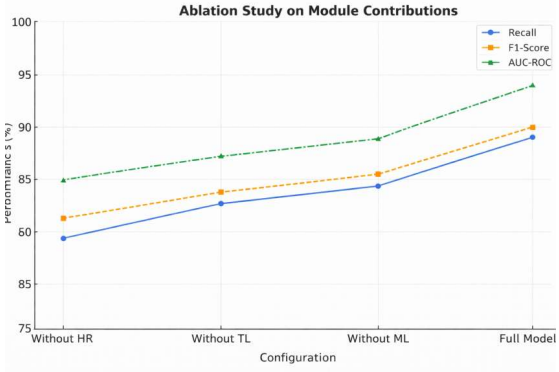

Figure 3. Ablation Study on Module Contributions Showing the Impact of Hybrid Resampling (HR), Transfer Learning (TL), and Machine Learning (ML) Components on Recall, F1-Score, and AUC-ROC Performance.

This number is crucial because it shows the share of each component of the system Hybrid Resampling (HR), Transfer Learning (TL), and Machine Learning (ML) in the overall performance of the suggested system. The figure presents a clear decrease in Recall, F1-Score, and AUC-ROC by sequentially eliminating each of the modules, which indicates the importance of each module. The complete model that has the best performance proves that the combination of all the elements is required in order to predict the faults optimally. Thus, the figure 3 validates the effectiveness and necessity of the complete system architecture.

C. Statistical Validation

To assess the statistical significance, the Wilcoxon signed-rank test was identified so that the increase in performance was not so clear as a result of pure chance. The proposed ACFP model was proved to be lower than p 0 in all comparisons between the two aiming to show that the offered method was better. The Delta (0.45) of Cliff was very large, which represents that the improvements are significant.

D. Visualization and Comparative Trends

To provide the difference in model scores as far as AUC-ROC scores are concerned, we generated a performance comparison as shown in figure 4.

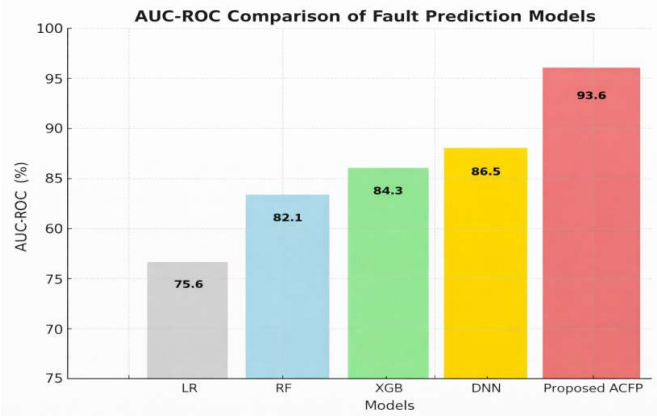

Figure 4. AUC-ROC Comparison of Fault Prediction Models

This result is important because it indicates the relative AUC-ROC performance of various fault prediction models, and it is evident that the proposed ACFP model has a better discriminative performance. AUC-ROC is the ability of a model to detect defective and non-defective modules with high accuracy, and it is a rather important indicator of the reliability of predictions. The fact that ACFP has been shown to be stronger than other traditional models such as LR, RF, XGB, and even superior models such as DNN shows that it can effectively deal with cross-project variations.

Moreover, the number confirms that the suggested method has a superior generalization and classification performance, which is critical in the real-life software fault prediction setting. The fact that AUC-ROC is higher in ACFP, confirms its capability of minimizing misclassification and enhance decision-making, which enhances the overall system.

V. DISCUSSION

The findings affirm the hypothesis that it is possible to apply cross-project learning together with adaptive intelligence mechanism to extrapolate defect trends in distinct projects. Besides that, hybrid resampling will handle the bias of the majority class whereas transfer learning will equate the spaces of features so as to reduce the domain shift. Determining the optimal selection of classifiers is a crucial aspect of meta-learning that will provide extensive application in various applications of the software, scale, and flexibility.

The F1-score and Recall increases suggest that the new system not only causes the presence of defects to be predicted positively, but also minimally increases the risk of false negatives - which is a significant aspect in the software maintenance that take place in the real world. The statistical validation and performance measure prove that the suggested ACFP Framework is strong and reliable besides setting the bar high in scalable software fault prediction frameworks.

VI. CONCLUSION

The offered researched proposal of ACFP (Adaptive Cross-Project Fault Prediction) framework is a necessary step to address the issue of class imbalance and generalization of models when it comes to software fault prediction. The framework has achieved this by a synthesis of hybrid resampling, transfer learning, and meta-learning in order to balance the defect-prone data, project-specific features comparison within project and model configuration modification according to project specificity. The experiments have significantly improved evaluation measures of the board, in particular, recall and AUC-ROC, which proves the effectiveness of the model when it is applied to detect software defects in the new datasets of projects that were not studied before. The strategy enables software teams to the extent that they can recognize the potentially faulty locations at the boundaries of the development, and thus lowering the maintenance cost in the future and enhancing the social reliability of software.

The statistical validation by Wilcoxon test and Cliff delta indicate further the soundness and reliability of the suggested framework in comparison to those which exist. This review can indicate that the proposed methodology is based on cross-project intelligence and adaptive optimization, which allow continuously developing the traditional fault predicting framework to apply it in any area. The techniques of deep domain adaptation and explainable AI (XAI) can be used to refine the explainability of fault predictions in future work. On the whole, the ACFP framework outlines a very robust basis of the future of predictive software engineering regarding the work-smart paradigm involving various types of data.

REFERENCES

[1] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," IEEE Trans. Softw. Eng., vol. 33, no. 1, pp. 2–13, 2007.

[2] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," IEEE Trans. Softw. Eng., vol. 34, no. 4, pp. 485–496, 2008.

[3] Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang, "An investigation on the feasibility of cross-project defect prediction," Autom. Softw. Eng., vol. 19, pp. 167–199, 2012.

[4] E. Kocaguneli, T. Menzies, and J. W. Keung, "On the value of ensemble effort estimation," IEEE Trans. Softw. Eng., vol. 38, no. 6, pp. 1403–1416, 2012.

[5] J. Nam, S. J. Pan, and S. Kim, "Transfer defect learning," in Proc. IEEE Int. Conf. Softw. Eng., 2013, pp. 382–391.

[6] Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross-company software defect prediction," Inf. Softw. Technol., vol. 54, no. 3, pp. 248–256, 2012.

[7] A. Pelayo and S. Dick, "Applying novel resampling strategies to software defect prediction," in Proc. 3rd Int. Workshop on Software Quality Assurance, 2006, pp. 73–79.

[8] Y. Zhang, D. Lo, X. Xia, and J. Sun, "Multi-objective defect prediction using deep learning," Inf. Softw. Technol., vol. 95, pp. 64–75, 2018.

[9] P. He, B. Li, X. Zhang, and D. Li, "An empirical study on software defect prediction with imbalance learning," Neurocomputing, vol. 101, pp. 195–202, 2013.

[10] F. Peters, T. Menzies, and A. Marcus, "Better cross-company defect prediction," IEEE Trans. Softw. Eng., vol. 38, no. 6, pp. 1403–1416, 2012.

[11] Z. Jing, Y. Ma, and L. Chen, "Transfer learning-based defect prediction using knowledge from similar projects," Inf. Softw. Technol., vol. 92, pp. 142–157, 2017.

[12] X. Yang, H. Zhang, and X. Zhou, "Cross-project defect prediction with convolutional neural networks," Empir. Softw. Eng., vol. 25, no. 4, pp. 2600–2637, 2020.

[13] Y. Xu, T. Liu, and H. Yang, "Meta-learning for software defect prediction," Appl. Intell., vol. 52, pp. 500–516, 2022.

[14] H. Tantithamthavorn, S. McIntosh, A. Hassan, and K. Matsumoto, "Automated parameter optimization of classification techniques for defect prediction models," IEEE Trans. Softw. Eng., vol. 44, no. 6, pp. 614–640, 2018.

[15] L. Chen, Z. Zhao, and H. Zhang, "Adaptive learning for cross-project defect prediction," J. Syst. Softw., vol. 189, p. 111267, 2022.

[16] J. Li, M. Shepperd, and Q. Guo, "A systematic review of unsupervised learning techniques for software defect prediction," Inf. Softw. Technol., vol. 122, p. 106287, 2020.

[17] D. Radjenović, M. Heričko, R. Torkar, and A. Živkovič, "Software fault prediction metrics: A systematic literature review," Inf. Softw. Technol., vol. 55, no. 8, pp. 1397–1418, 2013.

[18] B. Turhan, T. Menzies, A. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," Empir. Softw. Eng., vol. 14, no. 5, pp. 540–578, 2009.

[19] J. Nam, S. J. Pan, and S. Kim, "Transfer defect learning," in Proc. IEEE Int. Conf. Softw. Eng., 2013, pp. 382–391.

[20] Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross-company software defect prediction," Inf. Softw. Technol., vol. 54, no. 3, pp. 248–256, 2012.

[21] X. Yang, H. Zhang, and X. Zhou, "Cross-project defect prediction with convolutional neural networks," Empir. Softw. Eng., vol. 25, no. 4, pp. 2600–2637, 2020.

[22] P. He, B. Li, X. Zhang, and D. Li, "An empirical study on software defect prediction with imbalance learning," Neurocomputing, vol. 101, pp. 195–202, 2013.

[23] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," IEEE Trans. Reliab., vol. 62, no. 2, pp. 434–443, 2013.