Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.7749985

ENSEMBLE MODEL FOR SOFTWARE DEFECT PREDICTION USING METHOD LEVEL FEATURES OF SPRING FRAMEWORK OPEN SOURCE JAVA PROJECT FOR E-COMMERCE

Ramesh Ponnala
Research Scholar, UCE, Osmania University-Hyderabad and Asst. Professor, Department of MCA, Chaitanya Bharathi Institute of Technology (A), Gandipet, Hyderabad, Telangana-500075, India *e-mail: ramesh.ponnala@gmail.com, pramesh_mca@cbit.ac.in*

Dr.C.R.K. Reddy
Professor and Head, Department of CSE, Mahatma Gandhi Institute of Technology (MGIT), Gandipet, Hyderabad, Telangana-500075, India *e-mail*: crkreddy@gmail.com, crkreddy@mgit.ac.in

Abstract: This paper presents an ensemble model for software defect prediction using method-level features of a Spring Framework open-source Java project called Broadleaf Commerce. The proposed model uses a combination of 3 machine learning algorithms such as random forest, support vector machine using RBF Kernel, and LightGBM to predict the likelihood of software defects in the project. The method-level features considered for defect prediction include method complexity, method calls, and method length. The proposed ensemble model achieves a high ROC curve of 0.853 in defect prediction, outperforming the individual machine learning algorithms. The study demonstrates the effectiveness of using method-level features and ensemble models for software defect prediction in open-source Java projects. The results of the study can help software developers to identify potential defects and take corrective actions to improve software quality.

Keywords: Ensemble model, Software Defect Prediction, LightGBM, Random Forest, Software Quality

1. INTRODUCTION

Software defects are one of the major concerns for software developers as they can cause significant issues in software applications. Predicting software defects in advance can help developers identify potential issues and take corrective actions to prevent them. Machine learning-based defect prediction models have gained popularity in recent years for their ability to identify defects with high accuracy. However, individual machine learning algorithms may not always provide accurate results due to the complexity of software systems. Ensemble models can address this issue by combining multiple machine learning algorithms to improve the accuracy of defect prediction.

This paper proposes an ensemble model for software defect prediction using method-level features of a Spring Framework open-source Java project. Method-level features such as method complexity, method calls, and method length are used as predictors. The proposed ensemble model employs a combination of random forest, support vector machine, and

lightGBM algorithms to predict the likelihood of software defects in the project. The model's effectiveness is evaluated using various performance metrics, such as accuracy, precision, recall, F1-score, and ROC Curve. The results of the study demonstrate the effectiveness of using method-level features and ensemble models for software defect prediction in open-source Java projects. The proposed model can help software developers improve software quality by identifying potential defects in advance.
The remainder of the paper is organized as follows. Section 2 provides a brief overview of related work in software defect prediction. Section 3 describes the dataset used in the study and the method-level features considered for defect prediction. Section 4 presents the proposed ensemble model and its evaluation. Finally, Section 5 concludes the paper and discusses future work.

2. RELATED WORK

Software defect prediction(SDP) is a process to figure out bugs in software projects by considering various granularity software metrics as parameters. It is a bridge between software bugs and software metrics. There are many software metrics proposed for software defect prediction, McCabe Metrics, Halsted metrics [1], and Chidamber & Kemerer's (CK) [2] were the most widely used for software defect prediction for object-oriented software systems. Ramesh Ponnala et al. [3] studied a literature review on object-oriented software metrics proposed by many research scholars for a decade. Ramesh Ponnala et. al [4] studied the current state of the art of software defect prediction using various supervised and unsupervised machine learning algorithms and identified many researchers who worked with publicly available NASA or Promise repositories for defect prediction. Zhenyu et al. [5] proposed an ensemble model using Stacking algorithms with ANN, KNN, and Random Forest. They have worked with K-fold cross-validation techniques and got better results to compare with individual models. Yakub Kayode Saheed et al.[6] proposed several ensemble models such as The Catboost, LightGBM, XgBoost, boosted catboost, bagged logistic regression, boosted LightGBM, and boosted XgBoost. The proposed ensemble Catboost model gave an outstanding performance for all three defect datasets. Amal Alazba et al.[7] proposed the application of a stacking ensemble tree-based defect prediction model. To optimize the ensemble model they have used hyperparameters fine-tuned. Suneel Kumar Rath et al. [8] proposed a feature selection and support vector classifier-based software reliability model and validated the model using NASA Metrics Datasets.

3. DATASET DESCRIPTION

We have used the Bug Hunter Dataset constructed by Rudolf Ferenc et al. [9], in which 15 Open Source Java Projects were measured and constructed a database with various metrics as datasets in the format of comma-separated (CSV). Among these 15 open source java projects, Broadleaf Commerce is one of the Spring framework-based open-source java projects. Broadleaf Commerce is an open-source spring framework API for various types of E-commerce application development [10]. This dataset consists of method-level metrics with 75 features and 4709 records of data.
The dataset Broadleaf commerce dataset is preprocessed and subtracted version is collected from the bug hunter database. As there are 75 features, we applied the feature reduction

technique Principal Component Analysis using explained variance and reduced to 25 principal components based on the following figure 3.1 – a plotted graph concerning Explained Variance Vs Number of components based on the features. The amount of variation in a dataset that can be statistically explained by each of the principal components (eigenvectors) produced by the principal component analysis (PCA) approach is known as an explained variance.

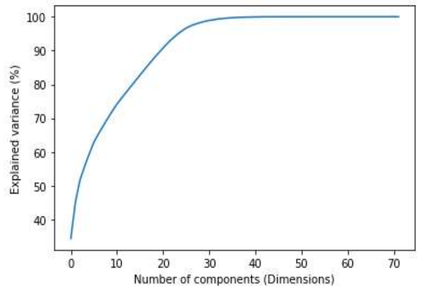
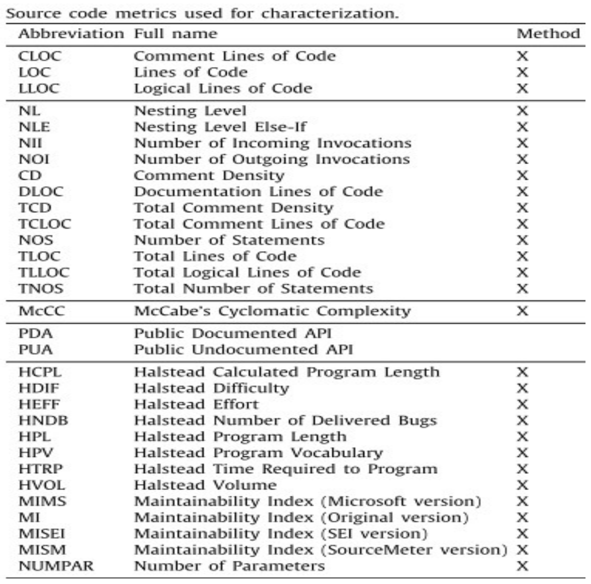


Figure 3.1 Explained Variance Vs Number of components

The following table gives some of the lists of source code method-level metrics used in the proposed model.

Table I Sample of Source Code Metrics at Method Level Granularity [9]



Source code metrics used for characterization.

Abbreviation	Full name	Method
CLOC	Comment Lines of Code	X
LOC	Lines of Code	X
LLOC	Logical Lines of Code	X
NL	Nesting Level	X
NLE	Nesting Level Else-If	X
NII	Number of Incoming Invocations	X
NOI	Number of Outgoing Invocations	X
CD	Comment Density	X
DLOC	Documentation Lines of Code	X
TCD	Total Comment Density	X
TCLOC	Total Comment Lines of Code	X
NOS	Number of Statements	X
TLOC	Total Lines of Code	X
TLLOC	Total Logical Lines of Code	X
TNOS	Total Number of Statements	X
McCC	McCabe's Cyclomatic Complexity	X
PDA	Public Documented API	
PUA	Public Undocumented API	
HCPL	Halstead Calculated Program Length	X
HDIF	Halstead Difficulty	X
HEFF	Halstead Effort	X
HNDB	Halstead Number of Delivered Bugs	X
HPL	Halstead Program Length	X
HPV	Halstead Program Vocabulary	X
HTRP	Halstead Time Required to Program	X
HVOL	Halstead Volume	X
MIMS	Maintainability Index (Microsoft version)	X
MI	Maintainability Index (Original version)	X
MISEI	Maintainability Index (SEI version)	X
MISM	Maintainability Index (SourceMeter version)	X
NUMPAR	Number of Parameters	X

4. PROPOSED ENSEMBLE MODEL & EVALUATION

We worked with the following machine learning models individually using the Datiku software tool [11] to build models using the feature reduction technique - Principal Component Analysis with 25 components.

- Logistic Regression
- Random Forest
- Support Vector Machine with RBF Kernel
- LightGBM

Then we combined Random Forest, SVM, and LightGBM to build an ensemble model using the Logistic Stacking approach. In machine learning, stacking refers to the design of a machine learning ensemble that determines the best way to combine each of the models in an ensemble to maximize its performance. Machine learning models generate relationship functions by mapping inputs to outputs. An important aspect of stacking is learning the relationship between the prediction result of each ensemble model and the actual value of out-of-sample predictions. The following figure depicts the workflow of the proposed ensemble model using Logistic Stacking [12].

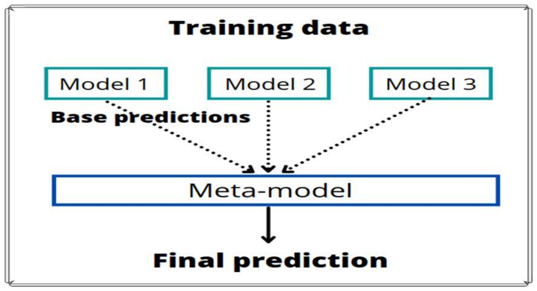


Figure 4.1 Generic Workflow of Ensemble Model

The Steps to be followed in the proposed ensemble model using the stacking technique can be summarized as:

Step 1: 5-Fold cross-validation is used to separate the data set into 5-Folds.

Step 2: Train multiple independent base models to each fold while holding one out.

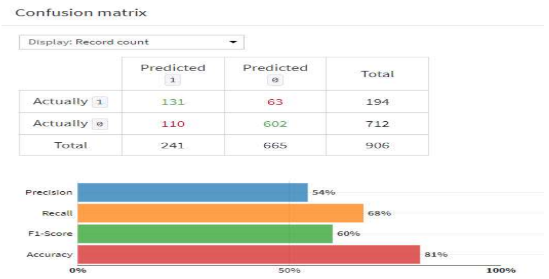
Step 3: The base model can be used to predict the hold-out fold

Step 4: The above three steps should be repeated 5 times to obtain out-of-sample predictions for all 5 folds.

Step 5: Provide the meta-model i.e. Logistic Regression with all of the out-of-sample predictions as features (training data).

Step 6: Predict the ultimate result using the meta-model.

The following Figure 4.2 represents the confusion matrix of the proposed ensemble model


Confusion matrix

Display: Record count

	Predicted 1	Predicted 0	Total
Actually 1	131	63	194
Actually 0	110	602	712
Total	241	665	906

Figure 4.2 Confusion Matrix of Ensemble Model

The following table summarizes the individual machine-learning models along with the proposed ensemble model scores concerning the Broadleaf commerce dataset.

Table I Summary of Machine Learning Model Scores

Algorithm	**ROC AUC**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Logistic Regression with PCA	0.74	0.67	0.37	0.72	0.48
Random forest with PCA	0.84	0.85	0.68	0.54	0.60
SVM with PCA	0.76	0.70	0.38	0.69	0.49
LightGBM with PCA	0.82	0.81	0.56	0.62	0.59

Proposed Ensemble Model	**0.853**	**0.81**	**0.54**	**0.68**	**0.60**

We observe that among the 4 individual models, Random forest and LightGBM performs better and the visual comparison of 4individual models is plotted in the following figure 4.3

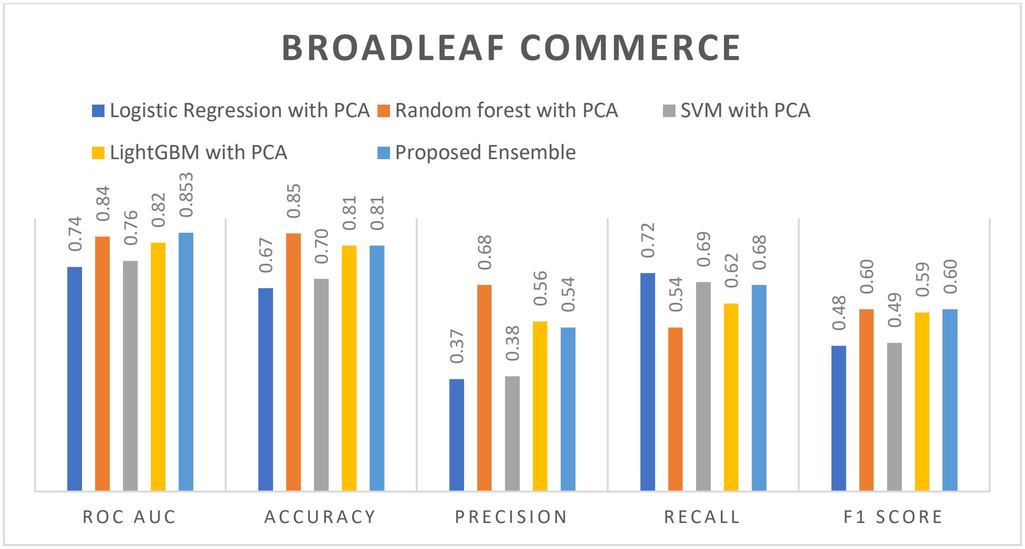


Figure 4.3 Comparison of Metrics with Proposed Ensemble Model with individual models

5. CONCLUSION AND FUTURE SCOPE

The open-source Java Spring Framework project's ensemble models for predicting software defects have shown good results using Logistic Stacking. More accuracy and reliability in defect prediction have been achieved through the use of ensemble models that aggregate the predictions of separate models. With proposed Ensemble, which is a combination of Random forest, SVM with RBF Kernel, and LightGBM have got a ROC curve of .853 and an accuracy of 81%, which is an excellent score in software defect prediction when compared with all individual models and other existing models.

Future research could be the of more advanced machine learning techniques, such as deep learning or reinforcement learning, to predict software defects. Furthermore, the research could expand to other software projects, not just limited to the Spring Framework. This would help generalize the approach and make it applicable to a wider range of software development projects.

REFERENCES

[1] "Maurice H. Halstead Elements of Software Science", [Online]. Available: https://dl.acm.org/doi/book/10.5555/540137#secAbs

[2] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object-oriented design," *IEEE Trans. Softw. Eng.*, vol. 20, no. 6, pp. 476–493, Jun. 1994, doi: 10.1109/32.295895.

[3] R. Ponnala and C. R. K. Reddy, "Object Oriented Dynamic Metrics in Software Development: A Literature Review," *Int. J. Appl. Eng. Res.*, vol. 14, no. 22, pp. 4161–4172, 2019, [Online]. Available: http://www.ripublication.com

[4] R. Ponnala and C. R. K. Reddy, “Software Defect Prediction using Machine Learning Algorithms: Current State of the Art,” *Solid State Technol.*, vol. 64, no. 2, 2021.

[5] Z. Yang, C. Jin, Y. Zhang, J. Wang, B. Yuan, and H. Li, “Software Defect Prediction: An Ensemble Learning Approach,” *J. Phys. Conf. Ser.*, vol. 2171, no. 1, 2022, doi: 10.1088/1742-6596/2171/1/012008.

[6] Y. K. Saheed, O. Longe, U. A. Baba, S. Rakshit, and N. R. Vajjhala, “An Ensemble Learning Approach for Software Defect Prediction in Developing Quality Software Product,” *Commun. Comput. Inf. Sci.*, vol. 1440 CCIS, no. October, pp. 317–326, 2021, doi: 10.1007/978-3-030-81462-5_29.

[7] A. Alazba and H. Aljamaan, “Software Defect Prediction Using Stacking Generalization of Optimized Tree-Based Ensembles,” *Appl. Sci.*, vol. 12, no. 9, 2022, doi: 10.3390/app12094577.

[8] S. K. Rath, M. Sahu, S. P. Das, and S. K. Mohapatra, “Hybrid Software Reliability Prediction Model Using Feature Selection and Support Vector Classifier,” *2022 Int. Conf. Emerg. Smart Comput. Informatics, ESCI 2022*, pp. 2–5, 2022, doi: 10.1109/ESCI53509.2022.9758339.

[9] R. Ferenc, P. Gyimesi, G. Gyimesi, Z. Tóth, and T. Gyimóthy, “An automatically created novel bug dataset and its validation in bug prediction,” *J. Syst. Softw.*, vol. 169, p. 110691, 2020, doi: 10.1016/j.jss.2020.110691.

[10] “Broadleaf Commerce”, [Online]. Available: https://www.broadleafcommerce.com/

[11] “Dataiku, Everday AI Tool” [Online]. Available: https://www.dataiku.com/

[12] Y. Lim, “Stacked Ensembles — Improving Model Performance on a Higher Level”, [Online]. Available: https://towardsdatascience.com/stacked-ensembles-improving-model-performance-on-a-higher-level-99ffc4ea5523