INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

RESTful Service based Software Defect Prediction using ML Algorithms

Ramesh Ponnala[1,2]*, **C R K Reddy**[3]

1 Research Scholar, UCE, Osmania University, Hyderabad, Telangana, India
2 Asst. Professor, Department of MCA, Chaitanya Bharathi Institute of Technology (A), Gandipet, Hyderabad, 500075, Telangana, India
3 Professor and Head, Department of CSE, Mahatma Gandhi Institute of Technology (A), Gandipet, Hyderabad, 500075, Telangana, India

OPEN ACCESS

Received: 05-06-2023
Accepted: 14-08-2023
Published: 15-09-2023

Citation: Ponnala R, Reddy CRK (2023) RESTful Service based Software Defect Prediction using ML Algorithms. Indian Journal of Science and Technology 16(34): 2789-2795. https://doi.org/10.17485/IJST/v16i34.1376

* **Corresponding author**.

ramesh.ponnala@gmail.com

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objectives: To present a suitable RESTful service-based software defect prediction approach that employs Machine Learning (ML) algorithms to identify software defects. **Methods:** The proposed approach is designed to provide a flexible solution for predicting software defects using various machine-learning techniques. It leverages RESTful web service-based class-level software metrics, including code complexity metrics, size metrics, coupling metrics, and cohesion metrics, and uses these metrics to train various ML models, such as Logistic Regression, Random Forest Classifier, LightGBM, XGBoost, and Support Vector Machines. **Findings:** We have proposed a correlation co-efficient method for feature selection and reduced it from 98 features to 25 features. With the granularity of class-level metrics of the RESTful service-based Elastic Search Engine's dataset, we achieved the highest F-measure score of 0.677 using the LightGBM Machine Learning model. The existing work was done using the 10-fold cross-validation and achieved an F-measure of 0.5817 using the Decision Table model. **Novelty:** Most of the existing works carried out by various researchers using publicly available NASA PROMISE datasets which were generated long ago on legacy programming languages and further no updates were taken into consideration. This could lead to data source bias, meaning the findings and models developed may not be representative of software systems from different domains or industries. The proposed work carried out is using a newly generated RESTful software defects-based dataset and publicly available: Bug Hunter Dataset. The Bug Hunter dataset aims to cover a wide range of projects and software systems from different domains and industries. This diversity allows researchers to develop defect prediction models that are more generalizable and applicable to real-world scenarios and specific organizations or domains. Apart from the original author, as of now, no one used this dataset for software defect prediction. In the proposed work we have used one of the Bug Hunter Datasets called Elastic Search Engine — a RESTful Service-based software. We have applied different feature selection methods and achieved the best results using the Correlation Coefficient technique and achieved the best F-Measure of 0.677 using LightGBM with a

hold-out validation approach whereas, in the existing work, the 10-Fold cross-validation technique was used and achieved 0.5817 as the highest F-measure using the Decision Table machine learning model. There is future scope for working with other Machine Learning Models for exhaustive comparison with the proposed model.

Keywords: Software Defect Prediction; Feature Reduction; Correlation Coefficient; Machine Learning; RESTful Service Software; LightGBM; Random Forest; SVM

1 Introduction

The NASA PROMISE repository dataset, may not cover the entire spectrum of software projects. It is biased toward certain programming languages, application domains, or project sizes. This lack of diversity could limit the generalizability of the findings to other software projects. The dataset lacks crucial contextual information about the software projects, such as the development process, team dynamics, or business requirements. Context plays a significant role in software defect prediction, and the absence of this information can limit the applicability of the results to real-world scenarios. The dataset's age may impact its relevance, as software development practices and technologies evolve. Models trained on older data might not be effective in predicting defects in more modern software projects. Imbalanced class distributions, where the number of defective instances is significantly smaller than the non-defective ones, are common in defect prediction datasets. This imbalance can affect the performance of machine learning algorithms, making it challenging to accurately predict defects. To deal with these problems, the proposed research work focused on RESTful Services - a modern web development, which enables the integration and communication of diverse software systems. RESTful services have become a standard in modern web development, enabling the integration and communication of diverse software systems. Using RESTful services as a basis for software defect prediction allows the analysis of defects in the context of distributed and interconnected software components. In this approach, data collection from RESTful services could be more challenging than traditional software defect prediction, as it involves monitoring the interactions between different services and extracting relevant features. Novel data collection and feature extraction techniques may have been developed to handle this specific scenario. RESTful services interact with various data sources and may produce different types of data (e.g., structured, semi-structured, unstructured). Dealing with heterogeneous data sources and effectively using them for defect prediction poses unique challenges, requiring specialized data processing techniques. Elastic Search Engine is a distributed, RESTful search and analytics engine capable of addressing a growing number of use cases. As the heart of the Elastic Stack, it centrally stores your data for lightning-fast search, fine-tuned relevancy, and powerful analytics that scale with ease. Software Defect Prediction (SDP) is an important aspect of Software Engineering to identify bugs, which is a crucial part of software quality assurance. Predicting bugs is a challenging task for the developer that requires the analysis of large amounts of software data, such as source code, log files, bug reports, and many other software artifacts. ML Models can be used to analyze this data and predict software defects with a good F1-Score.

Machine Learning algorithms can be applied to various types of source code metrics analysis, dynamic code analysis, and logs. These algorithms can be trained on various features extracted from these data sources. Several machine learning algorithms have been used for software defect prediction, including Logistic Regression, Decision Trees, Random Forests, Support Vector Machines, LightGBM, and Neural Networks. These algorithms are shown to get high accuracy in predicting defects, and their performance

can be improved by combining them into hybrid models. To significantly improve the software quality and reduce the cost of software development by enabling early defect prediction. Overall, the use of machine learning algorithms for software defect prediction has the potential to significantly improve software quality and reduce the cost of software development by enabling early detection of software defects. There is a requirement for a comprehensive evaluation of several machine learning algorithms to perform defect prediction. However, this paper proposed defect prediction models with few machine learning algorithms like Logistic Regression, Random Forest, LightGBM, and SVM, and the results are compared.

Rudolf Ferenc et al.[1] presented a bug hunter database based on open-source Java projects which are freely available containing source code elements and validated bug prediction using different machine learning algorithms and evaluated bug prediction with the F Measure over 0.74. Ramesh Ponnala et al.[2] studied various research articles relevant to software defect prediction using machine learning algorithms and summarized the current state of the art for the decade. Object-oriented dynamic metrics (OODM) are essential to measure the software application's efficiency. Ramesh Ponnala et al.[3] proposed a hybrid model to address class imbalance problems in SDP using ML Models. To balance the target variable various sampling techniques were applied and a balanced dataset was used for defect prediction using classification algorithms along with feature reduction techniques, and Random Forest with Oversampling techniques gave better results using two different datasets JUnit and Netty. A. Nageswara Rao Moparthi et al.[4] proposed a new hybrid model for defect prediction classification called Hybrid Phase Based Ensemble Classifier for the Pattern (HPBECPD). M.R. Ahmed et al.[5] proposed software fault prediction using 6 machine learning algorithms. They used a 10-fold cross-validation technique to evaluate the performance of ML models with 3 NASA repository datasets and results achieved 98-100%. Faseeha Matloob et. al[6] did systematic research on 46 papers and discovered that frequently employed hybrid models are random forest, boosting, and bagging. Zhenyu et al.[7] proposed an ensemble learning approach using Stacking algorithms with ANN, KNN, and Random Forest with K fold cross-validation techniques and proved that their proposed ensemble model gave better results compared to individual models. Santhosh Singh Rathore et al.[8] gave a method that dynamically selects learning techniques to predict the no. of defects in software and showed that it is the best prediction for the identified subset as a test dataset which is better than individual learning techniques and boosting bagging. Ramesh Ponnala et al[9]proposed an ensemble model with Random Forest, SVM, and LightGBM to predict defects using Spring Framework-based open-source Java project's dataset and achieved the highest ROC Curve of 0.853 and suggested working with more advanced techniques like Deep Learning model in software defect prediction.

2 Methodology

2.1 Data Collection

Z. Tóth et al. constructed Bug Hunter Dataset[10] source code metrics database of various large, popular, and publicly available open-source Java projects which are available on GitHub. The bug hunter dataset consists of 15 open source projects with static source code metrics, code duplication metrics, and code smell metrics, and the level of bugs considered are class, method, and file. In this paper, we use Elastic Search Engine, a popular RESTful search engine-based dataset with class-level metrics. After feature engineering, the authors have given a dataset of size 24994 rows and 98 features. As per the dataset the last feature is the Number of Bugs, as we are working to get a prediction of defect or not, we mapped the Number of bugs as a defect or not with binary values 0 and 1. We mapped the value 0 for zero number of bugs and the value 1 for more than zero as a number of bugs. So that we can have a target or dependent variable named defect. This reconstructed dataset can be used to train the machine learning classification models as binary classification models.

2.2 Approach

We have worked with an Elastic Search Engine-class level dataset with 3 different approaches for feature reduction like Correlation Coefficient, 10-Fold Cross-Validation with PCA (n_components=6), and based on explained variance PCA with 25 components. We used the following machine learning algorithms for the classification of defects.

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

Feature reduction using the correlation coefficient is a common technique to identify and select the most relevant features for a machine learning model. In this procedure, we calculate the correlation coefficient between each feature and the target variable "defect" and select the features with the highest correlation values. By setting a correlation threshold value of 0.8, we can control

the number of features selected for the reduced dataset. Adjusting the threshold allows us to balance between feature reduction and retaining meaningful information for accurate modeling.

The following Figure 1 depicts the overall workflow of the methodology used in this research work.

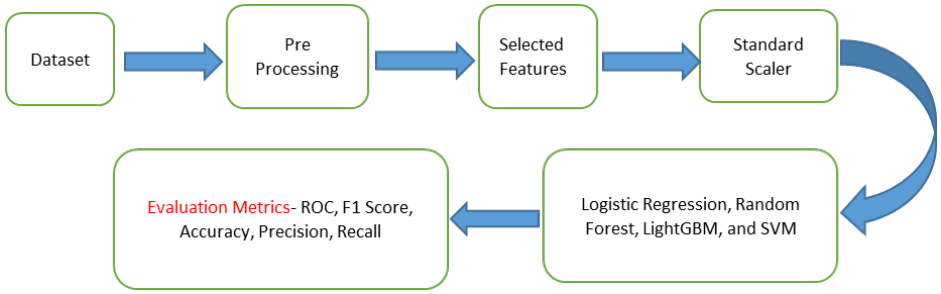

Fig 1. Research Workflow of Proposed Method

As per the above Figure 1, a dataset with 98 features will be pre-processed using a correlation coefficient and selects the top 25 features with the threshold value 0.8 to determine which features to select. Features with a correlation coefficient greater than this threshold will be considered relevant. Then these 25 features will be scaled to a range using StandardScaler normalization techniques to overcome outliers and trained by different models by splitting the main dataset as train and test split into 80:20 i.e., 80% of dataset samples are used for training and 20% of samples are used for testing. So that 20004 rows will be taken as train data and 4990 as test data. In future research work there is scope of working with more ML models and compare the present model results. The following Figure 2 shows hold-out validation approach of proposed ML models using the dataset.

Train & test sets

	LightGBM (ElasticSearch with Corre...	Random forest (ElasticSearch with ...	SVM (ElasticSearch with Correlatio...	Logistic Regression (ElasticSearch ...
Generated on	2023/02/24 10:43:08	2023/02/24 10:43:08	2023/02/24 10:43:08	2023/02/24 10:43:08
Train set rows	20004	20004	20004	20004
Test set rows	4990	4990	4990	4990

Fig 2. Hold-Out Validation Approach of ML Models

3 Results and Discussion

To compare the different machine learning models, we used accuracy, precision, recall, and F Measure metrics that are defined as follows:

- **Accuracy**: The accuracy is the proportion of correct predictions made by the model. It is the most commonly used metric for classification problems.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \; or \; Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions} \tag{1}$$

- **Precision**: Precision is the proportion of true positive predictions among all positive predictions made by the model. It measures how many of the predicted positive instances are actually positive.

$$Precision = \frac{TP}{TP+FP} \tag{2}$$

- **Recall**: Recall is the proportion of true positive predictions among all actual positive instances. It measures how many of the actual positive instances were predicted as positive.

$$Recall = \frac{TP}{TP+FN} \tag{3}$$

- **F Measure**: F1-score is the harmonic mean of precision and recall. It is used when you want to find the balance between precision and recall.

$$F\ Measure = \frac{2 * precision * recall}{precision + recall} \tag{4}$$

where TP (True Positive) is the number of classes that were predicted as defect and observed as defect, FP (False Positive) is the number of classes that were predicted as defect but observed as not defect, FN (False Negative) is the number of classes that were predicted as non- defect but observed as defect[(1)]. We evaluated the Elastic Search Engine dataset by applying correlation coefficient and, Principal Component Analysis to reduce the number of features from the 98 features. The following graph represents the correlation coefficient based on selected features using Random Forest and LightGBM Machine-Learning models.

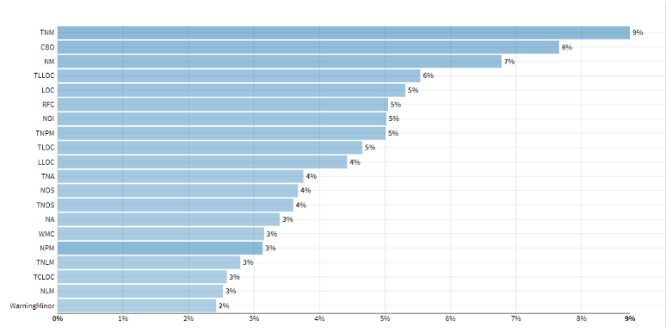

Fig 3. Variable importance of Random Forest

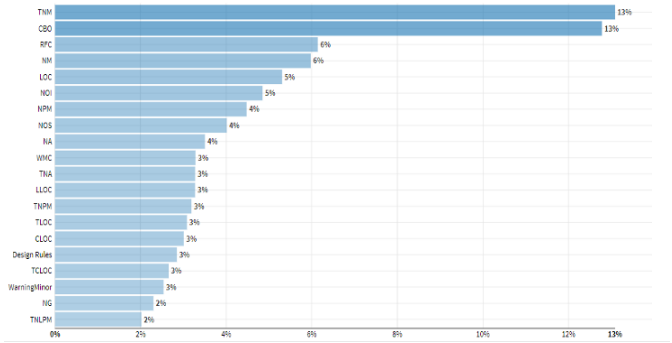

Fig 4. Variable importance of LightGBM

Figure 3 and Figure 4 reveal that the CBO, CLOC, Design Rules, LLOC, LOC, NA, NG, NLM, NLPM, NM, NOI, NOS, NPM, RFC, TCLOC, TLLOC, TLOC, TNA, TNLM, TNLPM, TNM, TNOS, TNPM, WMC, and WarningMinor features are selected as top 25 based on correlation coefficient by LightGBM ML Model. By considering these 25 features, we have trained the machine learning models and achieved the following evaluation metrics (Figure 5).

The following figures (Figure 6) depict the confusion matrix and relevant metrics in the form of a visualization.

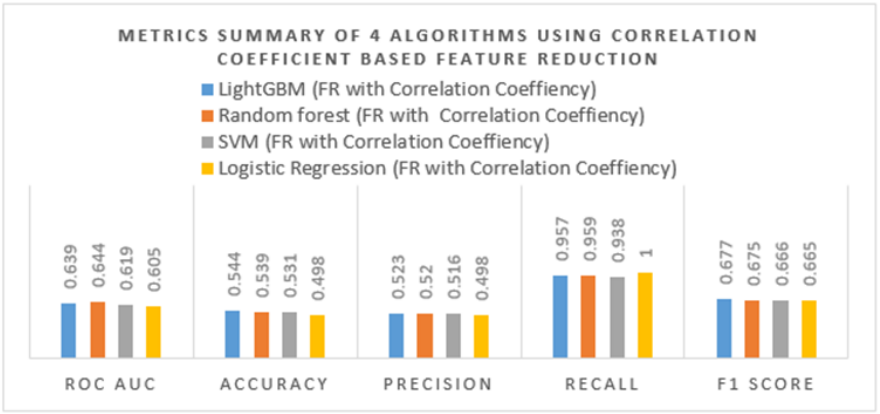

Fig 5. Metrics summary using Correlation Coefficient

a. Random Forest

b. Logistic Regression

c. LightGBM

d. SVM

Fig 6. Correlation Coefficient based Confusion Matrix and relevant metrics visualization of 4 ML models

From the Figure 6 confusion matrix, different metrics like Accuracy, Precision, Recall, and F1 Score are evaluated for the 4 different ML models[11]. When we compare this with the existing work carried out by Rudolf Ferenc et al.[1], in which they worked out with class-level metrics using the 10-Fold cross-validation technique and achieved 0.5817 as the highest F measure using the Decision Table machine learning model. In the proposed work we have worked with feature selection methods Correlation Coefficient with train-test split of 80:20 on selected models, Principal Component Analysis with 10-fold cross-validation and the Correlation Coefficient feature selection method gave the best result with 25 features. We have achieved 0.677 as the highest F- Measure using LightGBM and the second highest F-measure 0.675 using the Random Forest machine learning model. Apart from the correlation coefficient techniques we have worked with a 10-fold cross-validation technique with 6 principal components and Principal Component Analysis using explained variance with 25 principal components. From all these 3 approaches correlation coefficient with 25 features gave the highest F measure in defect prediction and all these approaches were carried out using Dataiku, an everyday AI tool free version.

4 Conclusion

In this research work, we worked with a publicly available Dataset: Elastic Search Engine Dataset, which is a RESTful service-based open-source project. We have worked with different feature selection methods like PCA, 10-fold cross-validation, and a correlation coefficient method, in which correlation coefficient feature selection reduced it from 98 features to 25 features and achieved the highest F measure score of 0.677 using the LightGBM machine learning model with the granularity of class-level metrics of Elastic Search Engine using hold-out Validation approach whereas the exiting work done by Rudolf et al.[(1)], they got 0.5817 with 10-fold cross-validation technique using Decision Table algorithm.

4.1 Limitations and Future Scope

With the Elastic Search Engine Dataset, apart from the original author, we are the first to work on it. The Proposed work is compared with the only one result, however there is a future scope of working with more ML models and compare the results for a more comprehensive analysis. In future work, there is a scope for working with RESTful service-based runtime log files, which will be useful to get the runtime metrics. There is another scope for working with the Prometheus tool to get run-time metrics and create datasets for defect prediction on modern state of art software applications.

References

1) Ferenc R, Gyimesi P, Gyimesi G, Tóth Z, Gyimóthy T. An automatically created novel bug dataset and its validation in bug prediction. *Journal of Systems and Software*. 2020;169:1–20. Available from: https://doi.org/10.1016/j.jss.2020.110691.
2) Ponnala R, Reddy CRK. Software Defect Prediction using Machine Learning Algorithms: Current State of the Art. *Solid State Technology*. 2021;64(2):6541–6556. Available from: http://solidstatetechnology.us/index.php/JSST/article/view/10794.
3) Ponnala R, Reddy CRK. Hybrid Model to Address Class Imbalance Problems in Software Defect Prediction using Advanced Computing Technique. In: 2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC), 04-06 May 2023, Salem, India. IEEE. 2023;p. 1115–1122. Available from: https://doi.org/10.1109/ICAAIC56838.2023.10141379.
4) Moparthi ANR, Geethanjali BN. Design and implementation of hybrid phase based ensemble technique for defect discovery using SDLC software metrics. In: 2016 2nd International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), 27-28 February 2016, Chennai, India. IEEE. 2016;p. 268–274. Available from: https://doi.org/10.1109/AEEICB.2016.7538287.
5) Ahmed MR, Ali MA, Ahmed N, Zamal MFB, Shamrat FMJM. The impact of software fault prediction in real-world application: An automated approach for software engineering. In: ICCDE '20: Proceedings of 2020 6th International Conference on Computing and Data Engineering. 2020;p. 247–251. Available from: https://doi.org/10.1145/3379247.3379278.
6) Matloob F, Ghazal TM, Taleb N, Aftab S, Ahmad M, Khan MA, et al. Software Defect Prediction Using Ensemble Learning: A Systematic Literature Review. *IEEE Access*. 2021;9:98754–98771. Available from: https://doi.org/10.1109/ACCESS.2021.3095559.
7) Yang Z, Jin C, Zhang Y, Wang J, Yuan B, Li H. Software Defect Prediction: An Ensemble Learning Approach. In: International Conference on Computer, Big Data and Artificial Intelligence (ICCBDAI 2021), 12/11/2021 - 14/11/2021 , Beihai, China;vol. 2171 of Journal of Physics: Conference Series. IOP Publishing. 2022;p. 1–7. Available from: https://doi.org/10.1088/1742-6596/2171/1/012008.
8) Rathore SS, Kumar S. An Approach for the Prediction of Number of Software Faults Based on the Dynamic Selection of Learning Techniques. *IEEE Transactions on Reliability*. 2019;68(1):216–236. Available from: https://doi.org/10.1109/TR.2018.2864206.
9) Ponnala R, Reddy CRK. Ensemble Model for Software Defect Prediction using Method Level Features of Spring Framework Open Source Java Project for E-Commerce. *Journal of Data Acquisition and Processing*. 2023;38(1):1645–1650. Available from: http://sjcjycl.cn/article/view-2023/1645.php.
10) Tóth Z, Gyimesi P, Ferenc R. A Public Bug Database of GitHub Projects and Its Application in Bug Prediction. In: ICCSA 2016: Computational Science and Its Applications;vol. 9789 of Lecture Notes in Computer Science book series. Springer International Publishing. 2016;p. 625–638. Available from: https://doi.org/10.1007/978-3-319-42089-9_44.
11) Shamrat FJM, Azam S, Karim A, Ahmed K, Bui FM, De Boer F. High-precision multiclass classification of lung disease through customized MobileNetV2 from chest X-ray images. *Computers in Biology and Medicine*. 2023;155:1–14. Available from: https://doi.org/10.1016/j.compbiomed.2023.106646.