An analysis of Software Maintainability Prediction Using Ensemble Learning Algorithms

Mothukuri JayaBharath
Department of Computer Science and Engineering
SRM University, AP
Amaravati , Guntur, India.
jayabharath_mothukuri@srmap.edu.in

Nandamuri Laasya Choudary
Department of Computer Science and Engineering
SRM University, AP
Amaravati , Guntur, India.
laasya_nandamuri@srmap.edu.in

Chebium Sai Pranay
Department of Computer Science and Engineering
SRM University, AP
Amaravati , Guntur, India.
saipranay_c@srmap.edu.in

Manasa Dhathri Praveenya
Department of Computer Science and Engineering
SRM University, AP
Amaravati , Guntur, India.
saipranay_c@srmap.edu.in

B Ramachandra Reddy
Department of Computer Science and Engineering
National Intitute of Technology
Jamshedpur, India.
brcreddy.cse@nitjsr.ac.in

***Abstract*—Software Maintenance is a long process and is the longest phase in the software development life cycle. Once the software is developed and delivered, maintenance plays an important role in the success of the software. The prediction of the effort required for software maintainability would result in effective management. In this paper, we used ensemble machine learning techniques to predict accurate effort required to maintain a software. Given two datasets, we implemented various machine learning algorithms to predict accuracy. The results show that the prediction using ensemble learning methods is more accurate due to less error. The least error in prediction is obtained while using Gradient Boost Classifier. Therefore, a more accurate prediction is obtained by using Gradient Boost machine learning algorithm.**

Keywords— Software Maintenance, Ensemble Learning, Adaboost, Gradient Boost, XGBoost, Cat Boost.

I. INTRODUCTION

A software project lifecycle consists of five different phases. They are planning, designing, development, testing, deployment and maintenance. The software project maintenance is the longest phase of a project life cycle and is the most costly phase as well. Maintenance of software requires a lot of effort. Predicting software updates and maintainability can help support and direct a few important aspects, such as the expenses of various projects and software-related decisions.. As a result, we can have control over the future software maintenance.

Recent research studies based on software maintainability prediction have different approaches to the prediction. None of them have proven to be best under all conditions. There are quite a lot of datasets and performance of the model may vary from dataset to dataset. Number of methods are available which are used to predict the software maintainability. Few of them are : ensemble methods, regression methods and so on. In order to offer a better result, ensemble methods use the capabilities of their constituent computer intelligence models (base learners) in the direction of a dataset. They are trustworthy and offer accurate predictions. The usefulness of ensemble approaches and the degree to which these ensembles improve or, in some situations, degrade prediction accuracy must therefore be supported by empirical data.

In this study, using ensemble methods and regression methods we predicted software maintainability. The intention of this research is to explore and evaluate different ensemble methods and to compare them against individual models and also among themselves. We used the QUES and UIMS datasets to forecast the software maintainability. By taking the metrics in these datasets we evaluated regression methods and ensemble methods. The metrics are Weighted Methods per Class (WMC), Depth of Inheritance (DIT), Number of Children (NOC), Response For a Class (RFC), Lack of Cohesion in Methods (LCOM), Message Passing Coupling (MPC), Number of Local Methods (NOM), Data Abstraction Coupling (DAC), and (SIZE1) traditional line of code, (SIZE2) total number of attributes and methods of a class[2][4]. We divided our data into two parts. 80% of the data is used for training and 20% of the data is used for testing[2][4].

II. RELATED WORKS

There are various research works that have implemented different methods to predict software maintainability. The link between object-oriented metrics and software systems' capacity to be maintained has been the subject of certain studies. There were significant links between the two. These metrics can also be used to predict the software maintainability.

Research by Quah and Thwin showed that broad regression neural networks outperform the Ward Network model for predicting software maintainability. According to Koten and Gray (2006), naive bayes models are more accurate at forecasting software maintainability for a single system than regression-based models. After further research, Elish and Elish (2009) concluded that the TreeNet technique can produce predictions with a higher degree of accuracy than the pre-existing prediction algorithms.

Recently, ensemble methods have become more well-liked and have improved single models' prediction accuracy in positive ways.(Braga et al. 2007; Sol-lich 1996). In software projects, many forecasts have been made using ensemble approaches. For instance, software project effort estimation, software fault prediction, and software reliability prediction. Additionally, they have been applied to a variety of real-world tasks, including face recognition, protein

structural class prediction, optical character recognition, and seismic signal classification. Except for the first work reported in Aljamaan and the later major extension of the same, published by Elish et al.[2][4], ensemble approaches have not, as far as we are aware, been employed for prediction of software maintenance.

This article compares the accuracy of prediction of software maintainability using regular regression methods and ensemble learning methods (homogeneous and heterogeneous).

III. PROPOSED METHOD

In this research, we have considered regression algorithms and ensemble methods to evaluate and compare them among themselves and against individual algorithms. We have considered the following regression techniques:artificial neural networks (ANN), linear regression and support vector regression (SVR).

For ensemble methods we have taken Adaptive Boosting (AdaBoost), Gradient Boost, Extreme Gradient Boosting (XGBoost), CatBoost. By applying these algorithms on both of the datasets UIMS and QUES, we will be able to know the more accurate and precise model compared to previous models.

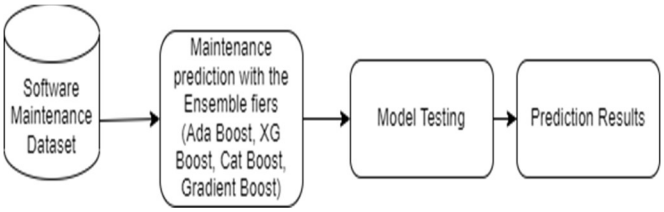

Fig. 1. Proposed Method: Block Diagram

A. The Ensemble Approach

The output of each of its unique constituent prediction models is used by a collection of computational intelligence models, each of which is given a specific priority level, to construct the final result with the aid of an arbitrator [2][4]. Individual prediction models in single-model ensembles are all of the same type, but each have a randomly generated training set (for instance, they may all be radial basis function networks). The single-model ensembles Bagging and Boosting are just two examples. Multimodel ensembles contain a variety of distinct constituent prediction models[4].

According to the arbitrator's design, Linear ensembles and nonlinear ensembles are additional categories for the multi-model ensembles[4]. The arbitrator in linear ensembles linearly averages, weights, or combines the outputs of the several models. No assumptions are made regarding the input provided to nonlinear ensembles[4]. The output of each individual prediction model is given to an arbitrator, a neural network, and a nonlinear prediction model that, after training, assigns the weights correctly[4].

The aforementioned ensemble gains from the fact that the errors of various prediction models vary throughout the used dataset partitions. This ensemble's goal is to select the best model for training from each dataset partition based on a set of criteria unique to that partition. The mean magnitude of relative error (MMRE) serves as the analysis's criterion. [4]

B. AdaBoost classifier:

An ensemble method called AdaBoost trains and shows outcomes as a series of trees. AdaBoost performs boosting, which requires chaining several weak classifiers together so that each weak classifier tries to reclassify data that was wrongly recognised by the weak classifier that came before it[6]. By combining weak classifiers to create a strong classifier, boosting achieves this. Decision trees used in boosting approaches are referred to as "stump" since each one tends to be a shallow model that does not overfit but might be biassed[6]. A specific tree has been trained to concentrate only on the flaws of the other tree. The weight of a sample that the prior tree misclassified will be increased so that the subsequent tree focuses on correctly categorising the misclassified sample[6]. AdaBoost performs badly in the presence of noise but excels in the presence of unbalanced datasets. The biggest drawback of Adaboost is how much longer it takes to train.

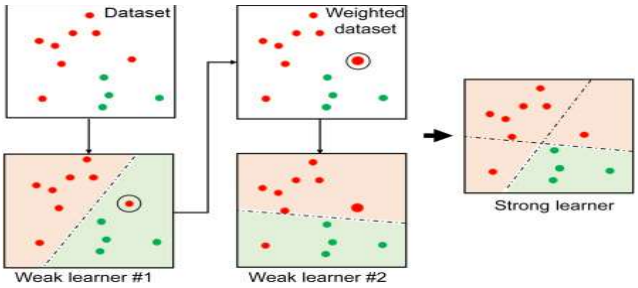

Fig. 2. AdaBoost Ensemble Method[6]

C. Gradient Boost Regression:

Gradient boosting is a machine learning technique that creates a prediction model in the form of an ensemble of weak prediction models for regression and classification tasks[7]. This method generalises the model by permitting the optimization of any differentiable loss function through the incremental building of a model. Gradient boosting essentially merges multiple weak learners into one strong learner iteratively[7]. For each subsequent weak learner, a new model is fitted to provide a more accurate estimate of the response variable[7]. The new weak learners have the highest correlation with the negative gradient of the loss function connected to the entire ensemble. Gradient boosting attempts to build a stronger prediction model by integrating a number of relatively weak prediction models.

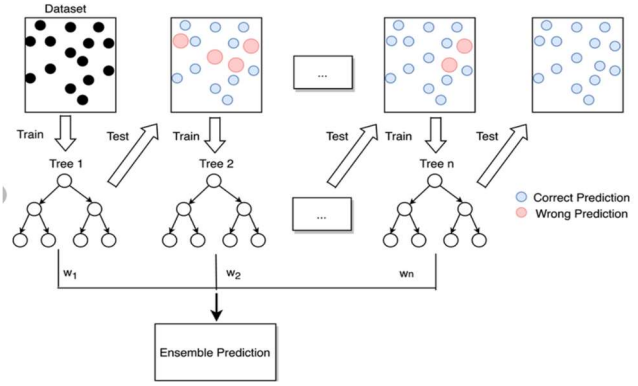

Fig. 3. Gradient Boosting approach[8]

D. XGBoost Regression:

XGBoost[9] is a gradient-boosted decision tree implementation. XGBoost models are primarily used in many Kaggle competitions. Decision trees are constructed using this method sequentially. Weights play an important role in XGBoost[9]. Every independent variable is assigned a weight before it is being fed into the decision tree that predicts results[9]. Variables that the first decision tree predicted inaccurately are given higher weight before entering into the second one. After that, through merging these distinct classifiers and predictors, a robust and accurate model is produced[9]. Regression, classification, ranking, and customised prediction problems can be solved with it.

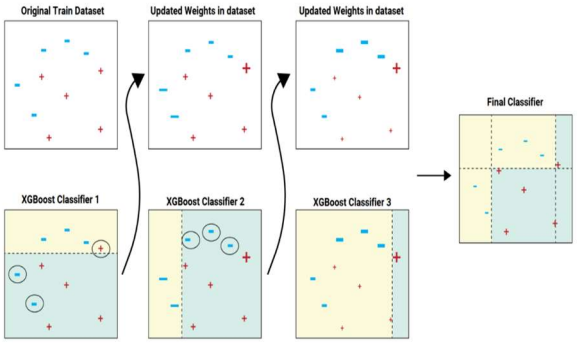

Fig .4. XGBoost method[10]

E. CatBoost Regression:

CatBoost is based on the principles of gradient boosting and decision trees[11]. Basic boosting objectives include successively integrating a large number of weak models (models that just marginally outperform chance), and then using greedy search to create a powerful, competitive prediction model. Gradient boosting fits the decision trees in a sequential manner, allowing the fitted trees to learn from the mistakes of earlier trees and so minimise errors[11]. New functions are continuously added to the current ones[11] until the selected loss function is no longer minimised.

Similar gradient boosting models are not used by CatBoost to build the decision trees.

Because CatBoost builds oblivious trees, which indicates that the trees are created by enforcing the restriction that all nodes at the same level test the same predictor under the same conditions, an index of a leaf can be determined using bitwise operations.

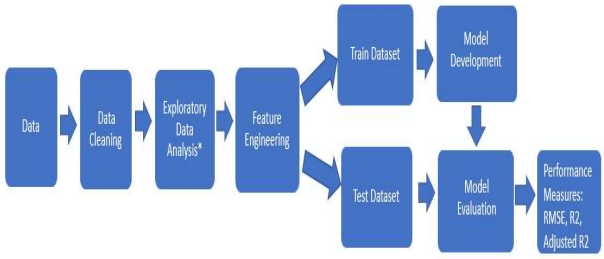

Fig. 5. CatBoost approach[12]

IV. Experimental Setup

The objective of this empirical evaluation is to determine how much more accurate the suggested ensemble technique would be at predicting software maintenance effort than individual methods[2][4].

A. Datasets

Two of Li and Henry's highly well-liked object-oriented software maintainability datasets, UIMS and QUES, were the subjects of our work. [2] [4] Since these datasets are freely available, our study is reliable, repeatable, and credible. A user interface management system's 39 classes provided the data for the UIMS dataset, while a quality assessment system's 71 classes provided the data for the QUES dataset. [2] [4] The two systems were put into action using Ada. Both datasets contain a total of eleven class-level measures, consisting of ten independent and one dependent measure. [2] [4]

The independent (input) variables consist of five Chidambar and Kemerer measurements (WMC, DIT, NOC, RFC, and LCOM), four Li and Henry metrics (DAC, NOM, SIZE2, and MPC,), and one traditional line-of-code metric. In Table 1, each statistic is succinctly explained. [2] [4]

The dependent (output) variable is the number of actual lines of code that were modified for each class during the course of a three-year maintenance cycle. the proxy measure of maintenance effort. [2] [4] A line change could be an insertion, deletion, or both. Addition and deletion are both considered when the content of a line changes. A discrete maintenance effort prediction model is created for each dataset because prior research on both datasets demonstrated that they are varied and have distinguishing qualities [2] [4].

TABLE I. FEATURES OF THE DATASETS

Metric	Description
WMC	Number of functions written in a program class
DIT	Class level in class hierarchy
NOC	Number of class's immediate subclasses
RFC	Sum of number of functions written in a class and count of functions that can be accessed by an object class due to inheritance
LCOM	The average percentage of functions in a class that are using each data element in the class subtracted from 100%
MPC	The count of messages a class has sent out
DAC	The count of instances that are declared within a class that belong to another class
NOM	The number of functions in a class
SIZE1	Number of LOC except comments
SIZE2	The total sum of the number of data attributes and the number of local functions that belong to a class

B. *Performance Measures*

Based on the relative error magnitude, we used de facto approved and widely accepted accuracy evaluation measures (MRE). These three variables are mean magnitude of relative error (MMRE), standard deviation of relative error, and level q prediction (Pred(q)) (StdMRE) [2][4].

where MREi is a normalised measure of the difference between the observed value (iactual)'s value (xi) and forecasted value (xi), The formula is as follows:

$$MRE = (|y_i - \hat{y}_i)|/y_i$$

MMRE is calculated as follows over a dataset of n observations:

$$MMRE = \frac{1}{n}\sum_{i-1}^{n}(|y_i - \hat{y}_i|/y_i)$$

Pred represents the proportion of observations having MREs that are lower than or equal to q [2][4].

$$Pred(q) = k/n$$

where n is the weighted total of all the observations in the dataset, and k is the number of observations whose MRE is less than or equal to a particular level q. The literature suggests that level q should not exceed 0.3. Therefore, we accepted that principle. [2][4].

V. Result Analysis

We used the 80:20 method of training and testing the model. We have taken two datasets QUES and UIMS into consideration and have predicted performance measures by taking the help of different algorithms.

The results of each algorithm's application to the QUES dataset are shown in Table 1. The individual algorithms that we have included in table 1 are ANN, SVR, and linear regression. We've determined the performance indicators MRE, RMSE, and Pred (0.3). It can be shown that, among all the models, linear regression fared better than each separate algorithm.

TABLE II. QUES DATASET PREDICTION RESULTS WITH ML

QUES dataset	MSE	RMSE	Pred(0.3)
Linear Regression	3770.43	61.40	0.067
SVR	829.67	28.80	0.33
ANN	2656.86	53.73	0.33

Table 2 shows the results obtained from performing the ensemble methods for the QUES dataset. The ensemble methods we have used in this research are AdaBoost, Gradient Boost, XGBoost and CatBoost regression. We have calculated the performance measures MSE, RMSE, Pred(0.3) for all these methods. Out of all the methods, Gradient Boost outperformed compared to all the other ensemble methods.

TABLE III. QUES DATASET PREDICTION RESULTS WITH ENSEMBLE

QUES Dataset	MSE	RMSE	Pred(0.3)
AdaBoost	3357.96	56.57	0.0
Gradient Boost	1094.26	26.01	0.46
XGBoost	3455.05	58.62	0.06
CatBoost	2272.26	47.66	0.26

A bar chart of the Pred(0.30) value obtained using each method is shown in Fig. 6. It is evident from the graph that the Gradient Boost regression produced the best Pred (0.30).

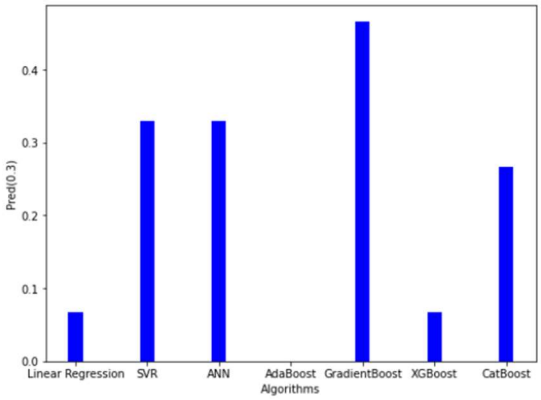

Fig. 6. Pred(0.3) for QUES Dataset

Fig. 7 shows a bar chart of the achieved RMSE values by each method for the QUES Data.

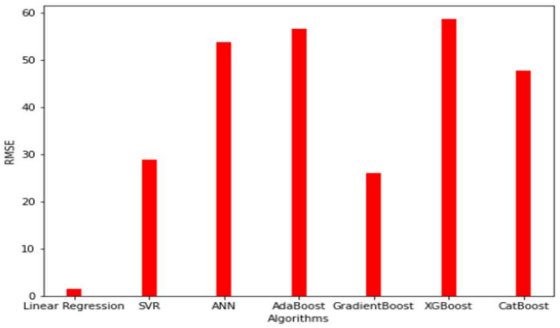

Fig. 7. RMSE for QUES Dataset

The outcomes from applying each technique to the UIMS dataset are shown in Table 3. The particular algorithms that we have included in table 3 are ANN, SVR, and linear regression. We have calculated the performance measures MRE, RMSE, Pred(0.3). Out of all the models it can be seen that linear regression outranked all the individual algorithms.

TABLE IV. UIMS DATASET PREDICTION RESULTS WITH ML

UIMS	MSE	RMSE	Pred(0.3)
LR	170.70	13.06	1.0
SVR	6731.32	82.04	0.25
ANN	8346.05	40.72	0.33

The outcomes of using ensemble methods on the UIMS dataset are presented in Table 4. In this study, we applied the ensemble methods AdaBoost, CatBoost regression, Gradient Boost, and XGBoost. We have calculated the performance measures MSE, RMSE, Pred(0.3) for all these methods. Out of all the methods, Gradient Boost outperformed compared to all the other ensemble methods.

TABLE V. QUES DATASET PREDICTION RESULTS WITH ENSEMBLE

UIMS	MSE	RMSE	Pred(0.3)
AdaBoost	514.52	22.90	0.5
Gradient Boost	975.0	30.57	0.5
XGBoost	138.44	21.22	0.62
CatBoost	524.25	22.89	0.62

A bar graph of each model's achieved Pred(0.30) value can be seen in Fig. 9. It is evident from the graph that the Gradient Boost has the largest Pred (0.30).

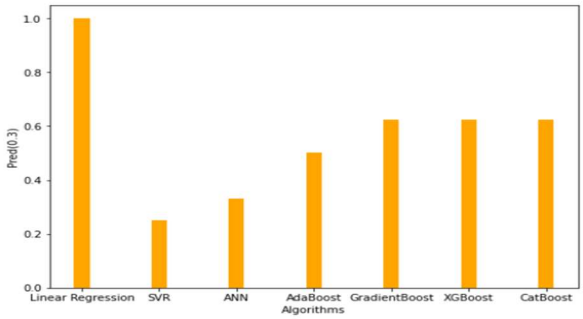

Fig. 8. Pred(0.3) for UIMS Dataset

A bar graph of each model's achieved Pred(0.30) value can be seen in Fig. 9.

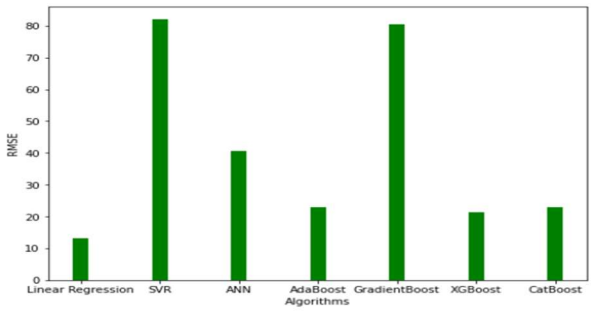

Fig. 9. RMSE for UIMS Dataset

There are several noteworthy findings when we look at the outcomes from both datasets. In the UIMS and QUES data, the Gradient Boost regression fared the best, first validating the theory that the effectiveness of the different prediction models may vary from dataset to dataset. Finally, Gradient Boost generally outperformed ensemble approaches in terms of accuracy.

VI. CONCLUSION

In this paper, we have presented different ensemble methods and individual algorithms for predicting the software maintenance. Initially we have applied basic machine learning (linear regression, SVR, and ANN) to predict the software maintenance and compared these results with the ensemble methods. Four well-known prediction models (AdaBoost, Gradient Boost, XGBoost, and CatBoost) were considered in ensemble approaches. Two publicly available software maintenance datasets were used to study and assess the effectiveness of prediction utilising ensemble methods (QUES and UIMS). After evaluating the datasets using the given algorithms and methods, the findings show that ensemble approaches are delivering more accurate and consistent predictions than individual algorithms hence it is more accurate. It is important to note that ensemble methods are computationally more expensive than the individual algorithms however, the advantages of the ensemble with respect to prediction accuracy. After evaluating the methods, we discovered that Pred(0.3) and MMRE values are having better results for Gradient Boost compared to other ensemble methods (Adaboost, XGBoost, CatBoost). Therefore, out of all the ensemble methods, Gradient Boost offers a more accurate prediction of software maintenance.

REFERENCES

[1] A. Kaur, K. Kaur, and Malhotra, R., "Soft computing approaches for prediction of software maintenance effort", International Journal of Computer Applications, Vol. 1(16), pp.69-75, 2010.

[2] M. O. Elish, H. Aljamaan, H., and I. Ahmad, "Three empirical studies on predicting software maintainability using ensemble methods", *Soft Computing*, Vol. 19(9), pp. 2511-2524, 2015..

[3] M. Riaz, E. Mendes, and E. Tempero, "A systematic review of software maintainability prediction and metrics", In *2009 3rd international symposium on empirical software engineering and measurement,* pp. 367-377,2009. IEEE.

[4] H. Aljamaan, M. O. Elish, and I. Ahmad, I, "An ensemble of computational intelligence models for software maintenance effort prediction", In *International Work-Conference on Artificial Neural Networks,* pp. 592-603, 2013. Springer, Berlin, Heidelberg.

[5] A. A. B. Baqais, M. Alshayeb, and Z. A. Baig, "Hybrid intelligent model for software maintenance prediction, 2014.

[6] S. Abirami, G. Kousalya, Balakrishnan and R. Karthick, "Varied Expression Analysis of Children With ASD Using Multimodal Deep Learning Technique", In Deep Learning and Parallel Computing Environment for Bioengineering Systems, pp. 225-243, 2022. Academic Press.

[7] En.wikipedia.org. 2022. Gradient boosting - Wikipedia. [online] Available at: <https://en.wikipedia.org/wiki/Gradient_boosting> [Accessed 6 May 2022].

[8] T. Zhang, W. Lin, A. M. Vogelmann, M. Zhang, S. Xie, Y. Qin, and J.C. Golaz, "Improving convection trigger functions in deep convective parameterization schemes using machine learning", Journal of Advances in Modeling Earth Systems, Vol. 13(5), p.e2020MS002365, 2021.

[9] S. Ramraj, N. Uzir, R. Sunil, and S. Banerjee, "Experimenting XGBoost algorithm for prediction and classification of different datasets", International Journal of Control Theory and Applications, Vol. 9(40), pp.651-662, 2016.

[10] T. Chen, and C. Guestrin, "Xgboost: A scalable tree boosting system", In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pp. 785-794, 2016.

[11] A.V. Dorogush, V. Ershov, and A. Gulin, "CatBoost: gradient boosting with categorical features support. arXiv preprint arXiv:1810.11363.

[12] Analytics Vidhya. 2022. *CatBoost | Predict Mental Fatigue Score using CatBoost.* [online] Available at: <https://www.analyticsvidhya.com/blog/2021/04/how-to-use-catboost-for-mental-fatigue-score-prediction/> [Accessed 6 May 2022].

[13] S.S. Rathore, "An exploratory analysis of regression methods for predicting faults in software systems. Soft Computing, Vol. 25, pp.14841-14872, 2021.