A Comparative Analysis and Prognosis of Software Functionality with Machine Learning Techniques

Dr Y Narasimha Rao
Professor and HOD, Department of Computer Science and Engineering,
QIS College of Engineering and Technology, Ongole.

Abstract— At different milestones in the software evolution process, software quality evaluation is a trivial task. This can be used to schedule performance assessment, quality management and project enhancement operations. Two techniques Linear Programming with Multiple Parameters (LPMP) and Quadratic Programming with Multiple Parameters (QPMP) for assessing the quality of software had been employed in the ongoing studies and researches. Several experts conducted research with Support Vector Machine (SVM), Neutral network (NN), C5.0 for quality assessment. These experiments had given poor and low results. In this research, by utilizing corresponding attributes of a multiple datasets, we fine-tuned prediction efficiency. In addition to employing a method of selecting a subgroup of relevant variables and variance matrix for getting greater and better results, we have applied different tests on latest approaches and accomplished good results for other predictive activities. Machine learning (ML) algorithms such as Logistic regression (LR), AdaBoost (AB), Random Decision Forest (RDF), Bagging Classifier (BC) and Classification Tree (CT) are executed on the data to forecast the software functionality, reliability and disclosed the association between the parameters of quality and production. The investigational outcomes proved that the measure of software quality can be well determined and assessed by ML techniques.

Keywords—*Software Reliability, Tree Boosting, AdaBoost, Machine Learning, Software Functionality.*

I INTRODUCTION

Machine learning techniques help us to investigate, generalize and predict large datasets. Machine learning is related closely to statistics and decision-making. Machine learning techniques are used for various purposes, such as weather forecasting, estimating the sales of a product, calculating the probability of a team winning in a match etc.

Many tech vendors need to develop and distribute quality software products in the specified time-frame and cost. However, forecasting the quality of early-stage applications would greatly help programmers in the management and quality assurance of applications, and would make the distribution of effort and resources more effective. Defects may arise in any stage of software evolution process starting form requirements analysis to deployment phase. So there is a need to perform assessment after completion of every milestone. The factors that measure the quality levels of software are number of defects per unit, security vulnerabilities, software process model, size of software etc. Among all the factors, number of defects per unit is considered as most important factor.

There are some non-functional qualities or characteristics to the standard of applications, such as durability, maintenance, accessibility, consistency and productivity. Even if many considerations remain, it is primarily the reliability and maintainability of the calculation of the

output of a program in operation. With a lower error or loss frequency, high quality applications should be accurate.

II RELATED WORKS

Ceran, A. A et al., (2020). [1] employed covariance matrix and feature selection method for predicting the software functionality.

Design flaws in the software affect the maintainability of the software. Thongkum, Pet al., (2020). [2] Employed Extreme Learning Machine (ELM) to predict design flaws of the software. They assessed 20 application packages, compared ML models and found out that one particular model outperformed all other models.

Based on number of defects in the software, two studies were introduced in the past for predicting the functionality of software. Both these studies used International Software Benchmarking Standard Group (ISBGS) dataset. ISBGS-10 data set released in January 2007 contains 106 attributes and 4,017 records.

Methods such as LPMP and QPMP were executed on ISBGS dataset in the first study [3] and finally the prediction accuracies were analyzed. The performance of LPMP and QPMP on the ISBSG-10 database is measured by employing k-fold cross validation technique. 11 features and 374 records were left in the ISBGS dataset after preprocessing. High and low are the two parameters used to indicate software quality level.

Software quality level is determined using the formula given in Equation 1.

$$S_Q = m_n+2*m_j+4*e_t \quad \text{(Eq.1)}$$

Where S_Q denotes Software Quality, m_n denotes number of minor defects, m_j denotes number of major defects and e_t denotes number of extreme defects.

In the second study [4], the software is classified as high or low by employing NN, SVM, and C5.0 classifiers. If the number of minor defects is not more than 10, extreme defects don't exist, and number of major defects is not more than 1, then the quality level of software is assumed as high class. The remaining cases are assumed as low class. 53 features and 746 projects were left in the ISBGS dataset after preprocessing.

To predict the software quality, Rashid et al. [5] employed experience based ML approach where the solutions to the prediction problems are solved using stored cases or past experiences. The parameters used for predicting the quality of the software are software evolution type, functional points count, degree of complexity, Lines Of Code (LOC), software developer skills and experience. Euclidian distance (ED) or The Manhattan distance (MD) is used to measure the deviation. The results were captured in the storage system when the estimated error is not more than 10%. The numbers of inputs which are received from the user are restricted to particular value. In order to predict accurate values, the values in the database must be close to each other.

Reddivari, S et al., (2019). [6] Conducted observations using 8 ML models to predict reliability and maintainability of software, and concluded that Random Forest classifier is the best performer than others with an Area under Curve (AUC) of more than 0.8.

Rana, R et al., (2015). [7] Proposed a systematic architecture adapted from ISO/IEC 15939 information model for the use of large scale software businesses of ML techniques for measurement and estimation of software quality.

Chandra, K et al., (2016). [8] Developed a prediction model to improve the quality of software by considering software versions' data points.

Prabha, C. L et al., (2020). [9] Used hybrid feature reduction scheme along with artificial neural networks to predict software defects.

Nalini, C et al., (2020). [10] Used code profiles and genetic neuro evolution algorithm to predict software defects.

Gu, Z., Wang, Jet al., (2020). [11] Suggested a consistency model for machine learning constructing energy systems that could be used in the quality improvement of production of the system.

Malhotra, R et al., (2020). [12] The research study examined academic papers conducted between January 1990 and January 2019, in which deep learning was used to test the estimation metrics for software efficiency. In this paper 20 different studies and 7 deep learning domain groups are listed in the software quality prediction metrics.

Immaculate, S. D, et al., (2019). [13] utilized three supervised machine learning algorithms to design and forecast the creation and usage of historical data based software glitches by classification regression, probabilistic classification, i.e. Naïve Bayes and classification trees.

P. Singh (2019). [14] performed detailed studies in order to analyse and engage with varied incarnations of classifiers on ongoing activities, including undersampling, oversampling and mixed approaches. Six approaches with six classifiers in 12 datasets are tested.

Malhotra, R et al., (2020). [15] Done comparisons on nine java based software open-source programmes using four usually-used extraction technologies from PROMISE repository. The findings of this analysis demonstrate that autoencoders are an efficient way of minimising effectively the dimensions of a data collection of programme defects.

M. Banga et al., (2019). [16] An analysis of software reliability models based on machine learning techniques was performed. Once the plenary work on defects caused during fault removal was reviewed, they had already suggested a new method, using machine learning methods, which were focused on detection of the most important parameters that impact software protection.

K. Tanaka et al., (2019). [17] Auto-sklearn has been tested by using software metrics from 20 free licensed software projects for intra-release defect foreclosure, as well as correlated auto-sklearn with various classifiers to forecast the number of flaws in software systems. Results revealed that auto-sklearn behaved in a similar way to random decision forest, which in previous studies is one of the better prediction models for defect prediction.

M. W. Thant et al., (2019). [18] Suggested a hybrid approach which is paired with the use of Minimal level-Redundancy-Maximum-Correlation (MRMC) function. Five NASA Metrics Data Program datasets were studied and test results demonstrated that the hybrid method with MRMC provided greater precision than Support Vector Machine.

Khan, F et al., (2020). [19] Used seven ML models along with artificial immune networks to predict defective components of software. The software bug prediction model findings have shown that the ML models with optimization of hyper parameters worked well than their default hyper parameters.

S. Rathaur et al., (2020). [20] Used an ML model i.e multiple linear regression to predict defect density in open source software. The predictor variables used are Source Lines of Code (SLOC), developers count, commits count and code churn. The rmality test was performed for the predictor variables and the correlation matrix was tested between the defect density of the free software and each of the predictor variables.

All the above mentioned studies used binary classification to predict software quality. By employing recent classification methods, considering size in terms of function points, we tried to improve accuracy levels of prediction models.

III MATERIALS AND METHODS

A. DATASETS

Evidence-Based Software Portfolio Management (EBSPM) dataset has 492 completed software projects from the Netherlands and Belgium, from 4 separate firms. Any of the EPSPM dataset's properties are seen in Table I. The recent release of the dataset on 24 Jul 2017 doesn't contain defect density value. The Defect Density value (D_d) was determined using the formula given below and D_d was introduced as a feature in both the datasets (EBSPM and ISBSG).

$$D_d = \text{defect} * 1000 / f_s.$$

Where f_s denotes functional size.

In compliance with the ISBSG Update 11(June 2009), 5052 projects are in action. The ISBSG data set has twenty features of primary level and 118 features of secondary level. Any of this data sample's features are seen in Table II.

B. DATA PREPROCESSING

The training accuracy is improved when the dataset was preprocessed effectively and efficiently. So, we removed and deleted rows with missing values and undefined values. The final summary of these two datasets were shown in Table III and Table IV. ISBSG Dataset were limited to 11 features and 1 target class, EBSPM Dataset has been limited to 10 and one target class following preprocessing. Further, as per the defect density values of the project in Table V and VI, we categorized the software quality indicator into four groups.

C. MACHINE LEARNING METHODS

Logistic regression (LR)

Method of logistic regression addresses questions of classification. It is designed to predict the possibility of a class or class object. Logistic regression approaches an s-formed curve under which the binary response variables estimate their characteristics. The translation from the logistic equation to the Ordinary Least Square-type equation obtains a dynamic optimized equation in this method. Below is the equation (1) resulting from the probabilistic method. P is the chance of Y=1 and 1-P the risk of obtaining Y=0.

$$ln\left(\frac{P}{1-P}\right) = c + dx \qquad \text{Eq. (1)}$$

P from the regression model can also be extracted. The regression function in Equation (2) measures the predicted likelihood of X with Y=1 for a given value.

$$P = \frac{\exp(c+dx)}{1+\exp(c+dx)} = \frac{e^{c+dx}}{1+e^{c+dx}} \qquad \text{Eq. (2)}$$

AdaBoost (AB)

Adaptive boosting has been successful in binary classification and makes the weak learner a strong learner by adjusting its weight.

Random Decision Forest (RDF)

Random Forest is an ensemble classifier which is supposed to be graded and regressed. It builds the number of classification trees on multiple data sub-samples and takes a minimum of

predictive precision and also tests model override. The classification of its performance is based on class mode and utilizes average trees for regression.

Bagging Classifier (BC)

Bagging classifier is an ensemble meta-estimator that matches each of the base classifiers in random subsets of the initial data set and then combines respective individual forecasts to form a final forecast (either by voting or an averaging).

Classification Tree (CT)

Classification tree is a tree formed in a recursive order where each node represents a potential decision with edges that indicate the possible pathway between nodes. Instance classification essentially parallels the direction from the tree's root to its leaves. The characteristics used for decision-making are selectively chosen to ensure a high degree of information gain.

Table I. Features of EBSPM Data

Project_ID	Organization	Short_Project_Description	Year_technical_go_live
323	3	Maintenance and enhancements project on an existing……	2012
480	3	Maintenance project on an existing Mobile application…	2016
482	3	Enhancements project on an existing CRM application…	2016
474	3	Enhancements project on an existing Internet application…	2016
297	1	Maintenance and new functionality release on ….	2012

Table II. Features of ISBSG Data

Project_ID	DataQuality Rating	UFPRating	YearofProject	CountApproach
10001	4	A	1998	5
10075	1	**B**	1994	3
10136	2	**B**	2004	3
10143	1	**A**	1998	3
10163	1	**A**	1994	4

TABLE III. ISBSG DATASET

Step	Attribute	Filter	Excluded Projects	Residual Projects

1	Defect Density	Null	4292	760
2	FP Standard	Other/Null/Not given	15	745

TABLE IV. EBSPM DATASET

Step	Attribute	Filter	Excluded Projects	Residual Projects
1	Defect Process	Null	222	270

TABLE V. EBSPM QUALITY LEVELS

TABLE VI. ISBSG QUALITY LEVELS

Quality Level	Defect Density	Excluded Projects
1	0-80	56
2	80-160	74
3	160-320	54
4	320-4875	86

Quality Level	Defect Density	Excluded Projects
1	0	273
2	0,5-20	246
3	20-80	173
4	80-4237	52

IV IMPLEMENTATION

After pre-processing the excel files have been interpreted and the particulars have been forwarded to a vector using Python function. There have been two parameters generated and one parameter has been given the objective attribute (Quality level) and another parameter was applied to other selected properties.

Next, the matrix of correlation is accomplished. Both data sets were shown to have an exceptionally high association with software consistency with the number of defects. The timeframe and cost of production of the software are likely to influence its consistency.

Secondly, the table of function significance indicates the target class's comparison to other classes. The most influential feature of the data collection is the number of errors, one of the variables used to determine consistency.

For implementation of the models, we used the Python scikit-learn library. The preparation and test details were split by a 33-percent ratio of 67 percent. Figure 1 indicates that a defect mechanism is highly significant in the EBSPM dataset, but its functionality affects software quality almost equally.

Cost and time both play a major role in estimating consistency. Figure 2 indicates, on the other hand, that in the ISBGS dataset defect quantity is again the most important characteristic. The remaining features are about the same and do not matter as many faults.

Software was graded as high-quality or low-quality in the previous immediately comparable two reports. This can lead, particularly when at frontiers, to wrong results. That's why the standard was split into four grades. Since we have four class types, prediction algorithms need to be used in multi-class predictions.

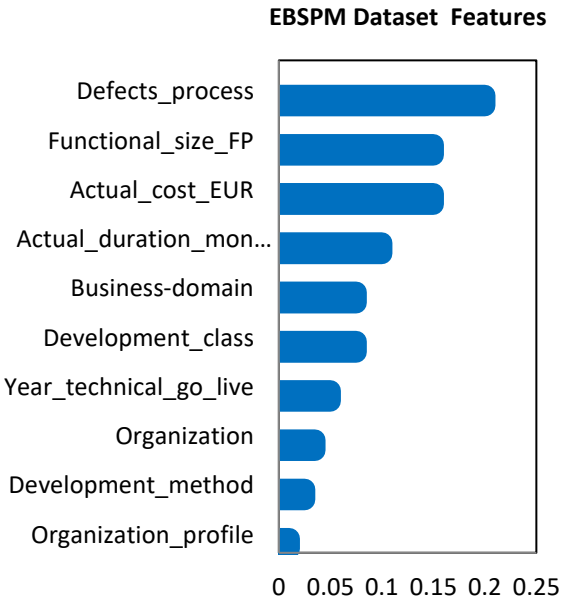

Fig. 1. Selected Attributes and their importance in EBSPM Dataset

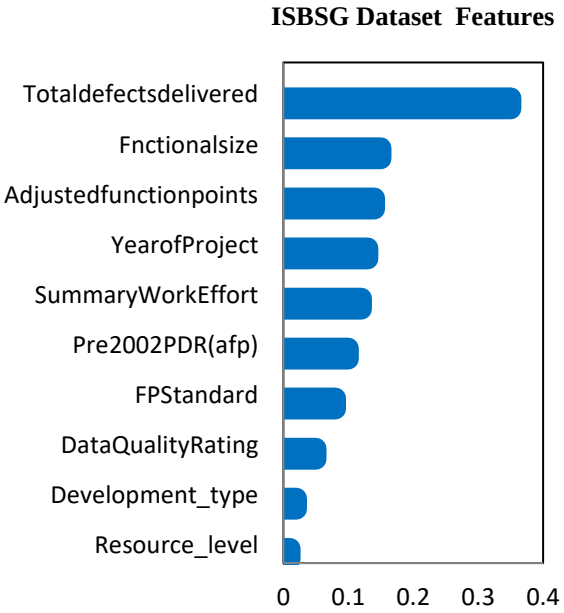

Fig. 2. Selected Attributes and their importance in ISBSG Dataset

V RESULTS AND DISCUSSION

The scikit-learn library algorithms were useful for our purposes, particularly in several class prediction problems. The findings of previous studies are shown in Table VII.

Table VIII and Table IX display the top five accuracies of the methods utilized. There were variations in precision between the interdependent coefficients in two datasets. Often, an essential element impacting precision is the gap in the variety of projects between two datasets.

We also evaluated classification techniques on two datasets using Scikit-learn library. Latest algorithms that stand up for multi-class classification have been researched by us. The exactness of these methods in EBSPM data set is 92.28% and in ISBSG data set is 92.22%

respectively. Appropriate level multiclass consistency estimation may be accomplished relative to previous strictly comparable tests. Graph showing results of current work is given in figure 3.

Table VII: Results of previous study methods

Technique	Accuracy
LPMP	61.70%
QPMP	66.90%
SVM	69.17%
NN	70.43%
C5.0	77.88%

Table VIII: Current work results on ISBSG DATASET

Technique	Accuracy
LR	69.92%
CT	89.43%
RDF	89.43%
AB	92.28%
BC	92.28%

Table IX: Current work results on EBSPM DATASET

Technique	Accuracy
AB	65.56%
RDF	66.67%
BC	67.78%
CT	67.78%
LR	92.22%

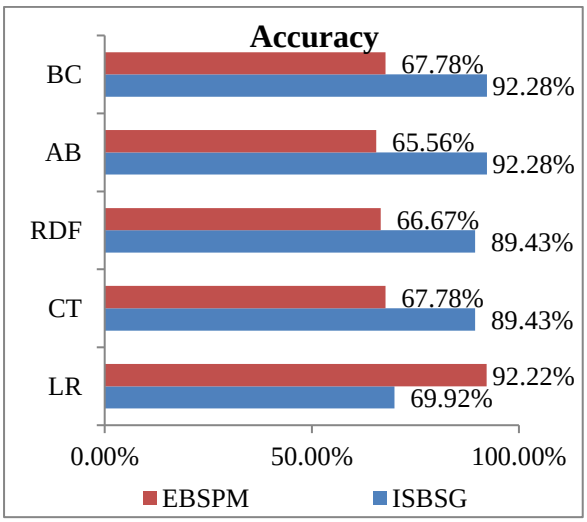

Figure 3: Graph showing results of current work

REFERENCES

[1] Ceran, A. A., & Tanriöver, Ö. Ö. (2020, June). An experimental study for software quality prediction with machine learning methods. In 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA) (pp. 1-4). IEEE.

[2] Thongkum, P., & Mekruksavanich, S. (2020, March). Design Flaws Prediction for Impact on Software Maintainability using Extreme Learning Machine. In 2020 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON) (pp. 79-82). IEEE.

[3] X. Wang, Y. Zhang, L. Zhang and Y. Shi, "A Knowledge Discovery Case Study of Software Quality Prediction: ISBSG Database," 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Toronto, ON, 2010, pp. 219-222.

[4] X. Wang, Y. Zhang, L. Zhang and Y. Shi, "A Knowledge Discovery Case Study of Software Quality Prediction Based on Classification Models: ISBSG Database," The 11th International Symposium on Knowledge Systems Sciences (KSS 2010), 2010.

[5] E. Rashid, S. Patnaik, and V. Bhattacherjee, "Software quality estimation using machine learning: Case-Based reasoning technique, " International Journal of Computer Applications, 2012.

[6] Reddivari, S., & Raman, J. (2019, July). Software Quality Prediction: An Investigation Based on Machine Learning. In 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI) (pp. 115-122). IEEE.

[7] Rana, R., & Staron, M. (2015, September). Machine learning approach for quality assessment and prediction in large software organizations. In 2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS) (pp. 1098-1101). IEEE.

[8] Chandra, K., Kapoor, G., Kohli, R., & Gupta, A. (2016, February). Improving software quality using machine learning. In 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH) (pp. 115-118). IEEE.

[9] Prabha, C. L., & Shivakumar, N. (2020, June). Software Defect Prediction Using Machine Learning Techniques. In 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184) (pp. 728-733). IEEE.

[10] Nalini, C., & Krishna, T. M. (2020, July). An Efficient Software Defect Prediction Model Using Neuro Evalution Algorithm based on Genetic Algorithm. In 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA) (pp. 135-138). IEEE.

[11] Gu, Z., Wang, J., & Luo, S. (2020, April). Investigation on the quality assurance procedure and evaluation methodology of machine learning building energy model systems. In 2020 International Conference on Urban Engineering and Management Science (ICUEMS) (pp. 96-99). IEEE.

[12] Malhotra, R., Gupta, S., & Singh, T. (2020, July). A Systematic Review on Application of Deep Learning Techniques for Software Quality Predictive Modeling. In 2020 International Conference on Computational Performance Evaluation (ComPE) (pp. 332-337). IEEE.

[13] Immaculate, S. D., Begam, M. F., & Floramary, M. (2019, March). Software bug prediction using supervised machine learning algorithms. In 2019 International Conference on Data Science and Communication (IconDSC) (pp. 1-7). IEEE.

[14] P. Singh, "Learning from Software defect datasets," 2019 5th International Conference on Signal Processing, Computing and Control (ISPCC), Solan, India, 2019, pp. 58-63, doi: 10.1109/ISPCC48220.2019.8988366.

[15] Malhotra, R., & Khan, K. (2020, June). A Study on Software Defect Prediction using Feature Extraction Techniques. In 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO) (pp. 1139-1144). IEEE.

[16] M. Banga, A. Bansal and A. Singh, "Implementation of Machine Learning Techniques in Software Reliability: A framework," 2019 International Conference on Automation, Computational and Technology Management (ICACTM), London, United Kingdom, 2019, pp. 241-245, doi: 10.1109/ICACTM.2019.8776830.

[17] K. Tanaka, A. Monden and Z. Yücel, "Prediction of Software Defects Using Automated Machine Learning," 2019 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Toyama, Japan, 2019, pp. 490-494, doi: 10.1109/SNPD.2019.8935839.

[18] M. W. Thant and N. T. T. Aung, "Software Defect Prediction using Hybrid Approach," 2019 International Conference on Advanced Information Technologies (ICAIT), Yangon, Myanmar, 2019, pp. 262-267, doi: 10.1109/AITC.2019.8921374.

[19] Khan, F., Kanwal, S., Alamri, S., & Mumtaz, B. (2020). Hyper-Parameter Optimization of Classifiers, Using an Artificial Immune Network and Its Application to Software Bug Prediction. IEEE Access, 8, 20954-20964.

[20] S. Rathaur, N. Kamath and U. Ghanekar, "Software Defect Density Prediction based on Multiple Linear Regression," 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2020, pp. 434-439, doi:10.1109/ICIRCA48905.2020.9183110.