IDENTIFIC ATION OF SOFTWARE EROSION USING LOGISTIC REGRESSION

Harinder Kaur[1], Raveen Bajwa[2]

[1]*PG Student., CSE., Baba Banda Singh Bahadur Engg. College, Fatehgarh Sahib, (India)*

[2]*Asstt. Prof., Deptt. of CSE/IT,Baba Banda Singh Bahadur Engg. College, Fatehgarh Sahib, (India)*

ABSTRACT

Typical number of components in a car is close to (1 million) and if each component is measured for its worthiness for assessment of the overall vitality of car and its life, we would have dataset consisting of large number of rows above 1 million, but in case of software where there are millions of lines of code, resulting large number of classes which need to be assured for overall health of the software for finding whether the projects are moving toward software rot or not, we would need a machine learning algorithm to handle such large dataset for pattern finding discovery having projects that learn on the fly. Therefore, in this research paper we have designed experiment for designing possible result for classifier for do so, the result shows in terms of TPR that the classifier we chosen Logistic Regression is performing best and is most appropriate to learn from dataset of metrics that influence software rot.

Keywords: Software Erosion, Machine Learning, Logistic Regression, Agile Development.

I. INTRODUCTION

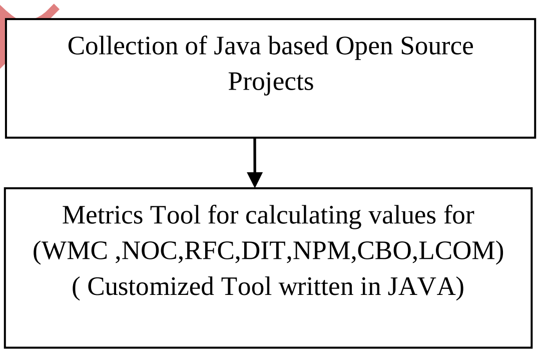
Software erosion in software products is a common problem and most of the software systems are affected by it. Software Erosion is refereed as a situation when due to forces of entropy, the application accumulates bugs, issue, incompatibilities with current environment due to multiple reasons including conflicts among the stakeholders and due to change due to emergence of new technologies in the immediate technical ecosystem. The software may suffer from irreparable and irreversible changes that may lead to abandonment of the software application. We have found that always, no matter how ambitious the intentions of the designers were, with the passage of time as these systems age, it becomes ever more difficult to make changes. Eventually it is more feasible to redesign and replace or at least refractor the software than it is to continue with the regular maintenance to extend the life of the existing design [1] .The Design decision taken early in evolution of system may conflict with requirements that need to be incorporated later in evolution[2]. Software Erosion is consequence of uncontrolled maintenance overtime which degrades quality of system. In that case it becomes mandatory to replace the existing one old system. However wholesale replacement of system from scratch is risky as it has great impact on technology, manpower and economic factors because replacement of system involve retraining of all users in order to understand new technology or may lack specific functionality of previous system as well as it imbalances the financial state of organization[3]. we would have dataset consisting millions of lines of code, resulting large number of classes which need to be assured for overall health of the software for finding whether the projects are moving toward software rot or not, we would need a machine learning algorithm to handle such large dataset for pattern finding discovery having projects that learn on the fly. Machine Learning is considered as a subfield of Artificial Intelligence and it is concerned with the development of techniques and methods which enable the computer to learn. In simple terms development of algorithms which enable the machine to learn and perform tasks and activities.

II. RELATED WORK

Threshold algorithm has been developed that helps to discriminate the metrics values into four categories(No rot, Low rot, Medium rot, High rot) for identification of degree for software rot ,that might occur in life cycle of software project been build, development and release based on agile development model which make use of dynamic range values for each metric [4]. Comparative study is given by author which concluding that logistic regression introduce less asymptotic error as the data grows [5]. This paper describes a different level in network traffic-analysis using an unsupervised machine learning technique, the flows are automatically classified by exploiting the different statistics characteristics of flow [6]. In one of another paper machine learning methods is applied for automatic Persian news classification by exert some language preprocess in Hamshahri dataset, and then extracted a feature vector for each news text by using feature weighting and feature selection algorithms then trained their classifier by support vector machine and K-nearest neighbor algorithms. The performance of KNN resulted better in comparison to other one [7].

III. METHODOLOGY

This section describe the various steps used for this complete process in fig.1. In this paper we have referred our previous paper for identification of software rot [4]. The


Collection of Java based Open Source Projects
Metrics Tool for calculating values for (WMC ,NOC,RFC,DIT,NPM,CBO,LCOM) (Customized Tool written in JAVA)

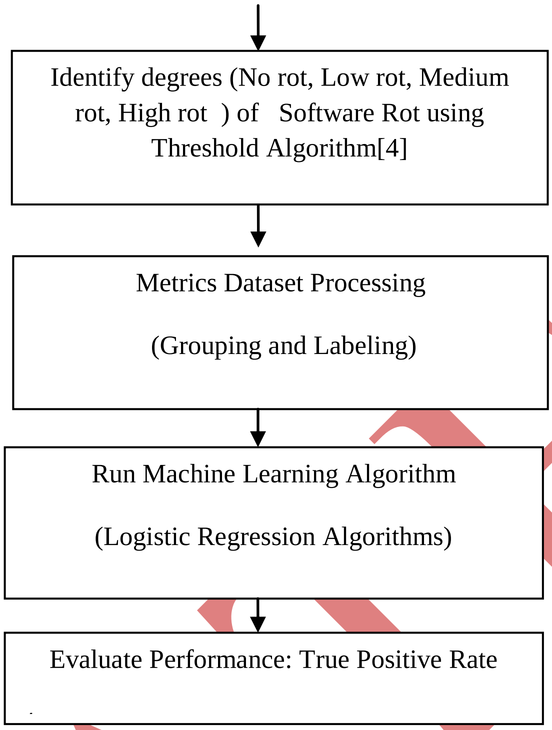

Identify degrees (No rot, Low rot, Medium rot, High rot) of Software Rot using Threshold Algorithm[4]
Metrics Dataset Processing
(Grouping and Labeling)
Run Machine Learning Algorithm
(Logistic Regression Algorithms)
Evaluate Performance: True Positive Rate

Fig 1: Process for Identification of Software Erosion.

IV. RESULTS AND DISCUSSION

In this section we have been showing the result by calculating the performance of logistic regression algorithm with various parameters results as mentioned below.

4.1 True Positive Rate

Table 1: True Positive Rate Values of Algorithm for Each Class

Class	True Positive Rate
No Rot	0.999
Low Rot	0.999
Medium Rot	0.999
High Rot	1
Average TRP	0.999

The true positive rate (TP) is the proportion of positive cases that were correctly identified, as calculated using the equation:

$$TP = \frac{d}{c + d}$$

d is the number of correct predictions that an instance is positive. c is the number of incorrect of predictions that an instance negative, Table-1 depicts the true positive rates for different classes (No rot, Low rot, Medium rot, High rot)for Logistic Regression algorithm. A high positive rate indicates that the classifier is capable of prediction which is very near to the specified criteria. Fig. 2 given above is showing graphical views of all four classes. The actual class group assumed by Threshold Algorithm [4] used for finding the degree of software rot.

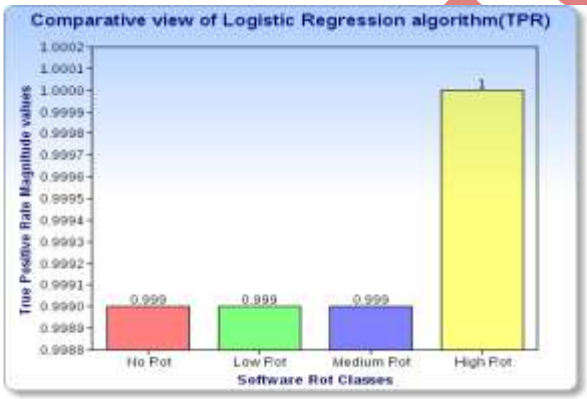


Fig. 2 Graphical Result for Logistic Regression Algorithm (TPR).

4.2 False Positive Rate

The false positive rate (FP) is the proportion of negatives cases that were incorrectly classified as positive.

Table-2 False Positive Rate Values of Algorithm for Each Class

Class	False Positive Rate
No Rot	0
Low Rot	0.001
Medium Rot	0
High Rot	0
Average FRP	0

False Positive Rate is calculated using the equation:

$$\mathrm{FP} = \frac{\mathrm{b}}{\mathrm{a} + \mathrm{b}}$$

Where b is incorrect number of predictions that an instance is negative and a is correct number of predictions that an instance is negative. Table 2 depicts the false positive rates for different classes (No rot, Low rot, Medium rot, High rot)for Logistic Regression algorithm. High value of false positive rate indicates that the algorithm is making large number of incorrect predictions and hence it is not reliable. The actual class group assumed by Threshold Algorithm [4] used for finding the degree of software rot. Fig. 3 given above is showing graphical views of all four classes.

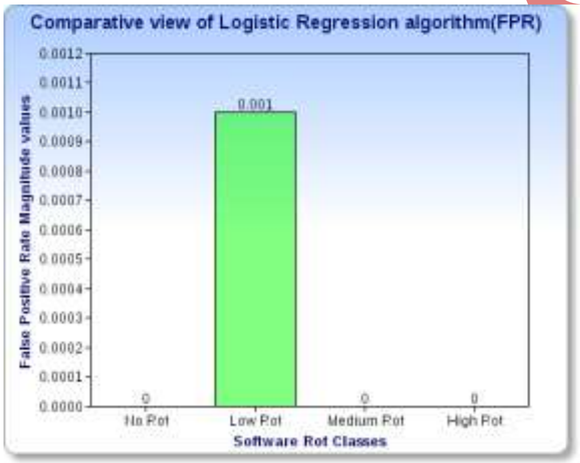


Fig. 3 Graphical Result for Logistic Regression Algorithm (FPR).

4.3 Precision

Table-3 Results of Precision Parameter of Algorithm for Each Class

Class	Precision
No Rot	0.999
Low Rot	0.995
Medium Rot	1
High Rot	1
Avg. Precision	0.99

Precision is the ratio of the number of relevant records matched to the total number of irrelevant and relevant records on the data set with each class. It is usually expressed as a percentage.

$$\text{Precision} = \frac{A}{A+C} X\ 100$$

A is Number of relevant records matched, C is Number of irrelevant records. Table 3 represent the Precision for different classes (No rot, Low rot, Medium rot, High rot)for Logistic Regression algorithm. Closer is the value of A/A+C to 1 higher the precision. Value closer to 1 indicates that no irrelevant records are being retrieved. Fig. 4 represents precision profile for different classifiers under investigation.

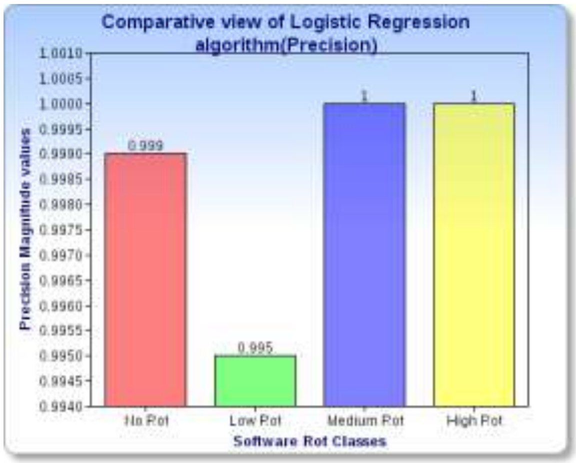


Fig. 4 Graphical Result for Logistic Regression Algorithm (Precision).

4.4 Recall

Table-4 Results of Recall Parameter of Algorithm for Each Class

Class	Recall
No Rot	0.999
Low Rot	0.998
Medium Rot	0.999
High Rot	1
Avg. Recall	0.999

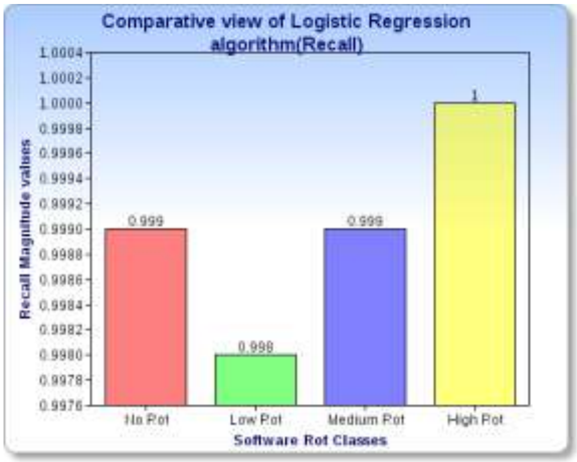


Fig. 5 Graphical Result For Logistic Regression Algorithm(Recall).

Table-4 showing the result of algorithm .Recall is the ratio of the number of relevant records matched to the total number of relevant records in the database. Fig. 5 depicts the Recall for different classes (No rot, Low rot, Medium rot, High rot) for Logistic Regression algorithm. It is usually expressed as a percentage.

$$\text{Recall} = \frac{A}{B} X\ 100$$

A is Number of relevant records matched, B is Number of relevant records not matched. Again closer the value A/B to 1 higher the recall.

V. CONCLUSION

In this research paper it is apparent that for each class (No rot, Low rot, Medium rot, High rot) that the Logistic regression algorithm is able to clearly categorize among the data pattern for each class. It may be attributed to the fact, that the nature of curve is log, which is used for finding the hyper plane making the rate of change high, hence we can say that a new contribution has been made in the field of software engineering to find the degree of software rot and by predicting the process of software rot degree by using logistic regression.

IV. FUTURE WORK

We suggest a combination of supervised and unsupervised machine learning algorithm must be used as unsupervised algorithm can be used to find pattern discovered in large dataset and supervised algorithms can be used to build highly accurate system of interpretation of software rot.

REFERENCES

Journal Papers:

[1] Er H.P.S Dhami and Anuj Kumar ,” Analysis of Software Design Erosion issues”, International Journal of Advanced Research in Computer Science and Software Engineering, ISSN: 2277 128X. Volume 3, Issue 7, July 2013.

[2] Van Gurp , Jilles, and Jan Bosch. "Design erosion: problems and causes." Journal of systems and software 61, no. 2: 105-119, 2002.(journal style)

[3] Pérez-Castillo, Ricardo, I. Garcia Rodrguez de Guzman, and Mario Piattini. "Diagnosis of software erosion through fuzzy logic." In *Computational Intelligence in Dynamic and Uncertain Environments (CIDUE), 2011 IEEE Symposium on*, pp. 49-56. IEEE, 2011.

[4] Kaur Harinder, and Raveen Bajwa. "Identification of Software Rot Using Range Control Limits." *International Journal* Scientific ResearchVolume-3 Issue 1,ID 02013725(2014).

[5] Jordan, A. "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes." *Advances in neural information processing systems* 2: (2002): 841.

Proceeding papers:

[6] Huixian, Liu, and Li Xiaojuan. "A novel traffic classification algorithm using machine learning." In *Broadband Network & Multimedia Technology, 2009. IC-BNMT'09. 2nd IEEE International Conference on*, pp. 340-344. IEEE, 2009.

[7] Farhoodi, Mojgan, and Alireza Yari. "Applying machine learning algorithms for automatic Persian text classification." In *Advanced Information Management and Service (IMS), 2010 6th International Conference on*, pp. 318-323. IEEE, 2010.

Books:

[8] Tutorial on Support Vector Machine (SVM) by Vikram aditya Jakkula, Washington State University Pullman 99164.

Proceeding paper:

[9] Tan, Yi, and Guo-Ji Zhang. "The application of machine learning algorithm in underwriting process." In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, vol. 6, pp. 3523-3527. IEEE, 2005.