Volume: 12 Issue:04 Pages: 4650-4655(2021) ISSN: 0975-0290

A Model Vector Machine Tree Classification for Software Fault Forecast Model (TSMO/TSVM)

First Author Maaz Rasheed Malik
Dept. of Information Communication Engineering, Guilin University of Electronic Technology, Guilin, China
Email: dr.maazmalik@outlook.com
Second Author Liu Yining
Dept. of Information Communication Engineering, Guilin University of Electronic Technology, Guilin, China
Email : ynliu@guet.edu.cn
Third Author Salahuddin Shaikh
School of Control & Computer Engineering, North China Electric Power University, Beijing, China
Email : engineersalahuddin@gmail.com

ABSTRACT

Many researchers have worked on the Software fault forecast model because the software fault forecast is very important in software development projects. In terms of the Software fault forecast model, earlier researchers have examined defective datasets models with the help of metrics and classification methods. Classification is assuming an exceptionally major job in the software fault forecast model, which is an important issue in data mining. The machine learning system as a finding way for the information securing or information extraction issue has examined it widely. The contribution to a classifier is a training data set of precedents, every one of which is labeled with a class name. Classification separates data tests into target classes. Software modules are categorized as defected models or not defected models by classification draws near. In Classification, class categories are known thus it is a supervised learning approach. In our research, software fault forecast datasets models are examined with the help of tree vector machine classification. Our proposed model is a tree vector machine, which is used for increasing the positive accuracy and efficiency of the software fault forecast model. We have used multiple tree classifiers for getting more accurate results and compare them with each other. During the analysis of the experiments, j48, random forest and random tree have increased their performance in accuracy as well as efficiency. However, the performance of REP Tree, Hoeffding Tree and Decision Stump is not so good at all measure rates. The experiment's analysis results showed that not every tree classifier could be good in all in measure unit.

Keywords - **Software, Fault Forecast, Classification, Defect prone, Support Vector Machine, J48, Random Tree.**

Date of Submission: Feb 12, 2021 Date of Acceptance: Feb 26, 2021

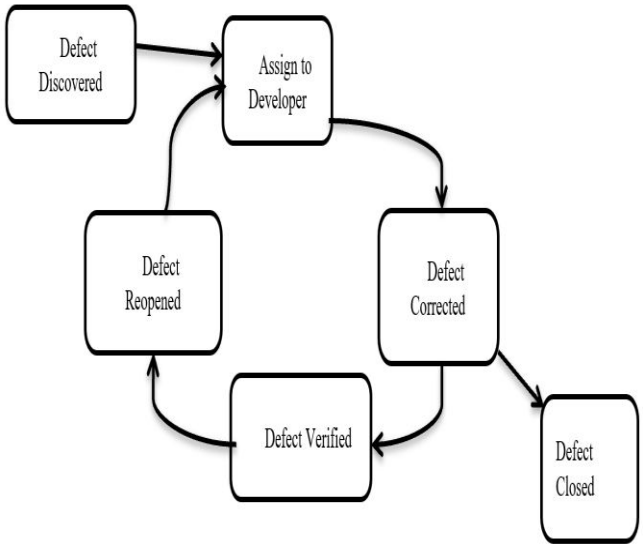
I. INTRODUCTION

Software fault forecast is a quality confirmation strategy in software building, where advanced methods (counting machine learning) are utilized to anticipate future deformities in PC programs. Such data can be utilized to help optimal endeavors and assets allocation in the software improvement ventures (for example to concentrate quality confirmation exercises on software classes, which are anticipated to be imperfection, inclined). As huge software systems are created over a time of several years, their structure will in general debase and it turns out to be progressively hard to comprehend and transform them.

Troublesome changes are unnecessarily exorbitant or require a too much long interval to complete. Day by day software efficiency and tools are developed in storage capacity and intricacy. The software forecast consistency assumes a fundamental job in the software expansion method. During the runtime of the software system, an error or bug is generated due to unambiguous and possible violations of security rules as well as wrong detail and unseemly advancement of setup which is known as a software bug. As indicated by software analyzing and anticipating imperfections are required for three fundamental purposes, right off the bat, for evaluating venture progress and plan deformity detection exercises for the task execution and also, for evaluating item quality and third in keep going to enhance the capacity and evaluate development routine for large organizations. In terms of software bug forecast, earlier researchers examined defective datasets models in the presence of particular metrics and classify the fault forecast models or prone models. The most significant models are that these prone models, number of bugs and imperfection prone are removed for enhancement of software worth, excellence and experiment analysis due to the update upcoming generation of the software. During the software bug forecast, there are two advantages, which are the enhancement of the experiment model through concentrating on defected modules and by recognizable proof the refactoring competitors which are anticipated as bug-inclined. To overcome the problem of software fault forecast model, genetic programming, fuzzy logic, neural network and so on multiple ways used. Probably metrics are used and studies for software fault forecast but defected datasets models before software free to assemble software fault forecast models, which are famous as

Volume: 12 Issue:04 Pages: 4650-4655(2021) ISSN: 0975-0290

supervised learning study. This is the first stage procedure approach utilized for software fault forecast and relies upon two stages: training and test stage; after handling the training stage, and their experiment results are known as Model. A testing data have to-do few executions with the help of this model. An unsupervised learning study is also developed during the software fault forecast when there are no past existing datasets models and a few different ways similar to grouping, which could be utilized.



Flow Chart 1: Software Fault Forecast Model Flow-Chart

II. Related Work

There are different kinds of ways for Software Fault Forecast models, for example, statistical way, machine-learning way, etc. In any case, the pattern as of late is moved from traditional statistical methods to machine learning methods. In the year 2006, a specialist named Zhou et al. depicted two types of research to defeat the issue of Software Fault Forecast. He observed logistic regression and machine learning methods, with the assistance of these two types of research he analyzed how object-oriented metrics and fault inclination are connected. In the year of 2003, Shuji et al. with his group showed one thought in regards to software fault forecast and they construct a model to detect imperfection adjustment exertion dependent on broadened affiliation rule mining. They characterized deformity-fixing exertion as a variable and fitting affiliation rule mining to treat with such factors. The information utilized is upheld by Japan's Ministry of Economy, Trade, and Industry (METI).

They use backing and certainty as evaluation factors. Their methodology expressed outcomes as a mean of redress exertion dependent on improvement level. For instance, deserts detected in coding and unit levels will be anything but difficult to address 7% of mean exertion when they are accompanying validation of info information.

In the year of 2002 Gray et al. was presented static code metrics. This metrics code utilized with an SVM classifier for an accumulation of modules in NASA REPOSITORY informational indexes. Rigorous et al. presented pre-processing steps that were applied to information preceding classification, including balancing the two classes (blemished or non-damaged) and removal of many rehashing occurrences. The SVM in this examination yielded a normal of 70% exactness on previously concealed information.

A comprehensive empirical investigation examining a wide range of groups of feature ranking strategies (rankers) including a couple of regular feature ranking ways, signal-to-commotion channel strategy, and others edge-based feature ranking procedures was presented by Wang et al. This examination utilized 16 real-world software estimation informational indexes of different sizes and constructed 13,600 classification models. Results showed that gatherings of a couple of rankers are compelling with other rankers and better than those of numerous rankers.

In the year of 2007, Mizuno and Kikuno announced that about their systems and they contemplated, Orthogonal Sparse Bigrams Markov models (OSB) are most appropriate to fault forecast. In the year of 2006, Bibi et al. revealed that software fault forecast can be defeated their issue with the utilization of Regression via Classification and did functions admirably.

In the year of 2005, Khoshgoftaar et al. detailed that utilizing of decision tree strategy can be viably grouped for those modules whose imperfection inclination is predicted as dubious. In the year of 2012, Hall et al. recommended that Naïve Bayes and Logistic regression strategies are work best for his analysis of the outcomes from 19 examinations.

III. Major Principal Classification Studies

Classification is assuming an exceptionally major job in the software fault forecast model, which is an important issue in data mining. The machine learning system as a finding way for the information securing or information extraction issue has examined it widely. The contribution to a classifier is a training data set of precedents, every one of which is labeled with a class name. Many attributes esteem characterizes each record. Attributes with discrete spaces are alluded to as categorical, while those with ordered areas are alluded to as numeric. The objective is to actuate a model or depiction for each class as far as the attributes. The model is used for classifying future records whose classes are obscure.

Further, Data classification is elaborated in the binary method, a first method is that all encoded set of data classes are explained. All database tuples are given details through attribute and evaluate by this model. A class label attribute is examined here where every tuple firm through attribute and have a relation with existing, class. The data tuples examined to construct the model altogether form the training data. This procedure is called machine learning.

There are two learning and one is supervised learning and the other one is unsupervised learning. In supervised learning, every training sample of the class label is offered. But on another side unsupervised learning where every training sample of the class label is not identified and a set of classes is also not identified in proceeds.

Volume: 12 Issue:04 Pages: 4650-4655(2021) ISSN: 0975-0290

Usually, attribute rule classification like rule classification and decision rules are always explained the learning model. These principles always used for superior comprehension of the database model and classified in future data models.

In the subsequent method, the classification is performed by the model then initially, the prescient precision of the model is assessed. The model is possible to evaluate for classification future data tuples when the accuracy of the model should be considered acceptable for the class label which is not identified.

Classification separates data tests into target classes. Software modules are categorized as defected models or not defected models by classification draws near. In Classification, class categories are known thus it is a supervised learning approach. There are two classification strategies: Binary and Multilevel. Parallel classification separates a class into two categories; faulty data and not-faulty data. While Multi-level classification is utilized when there are more than two classes. It separates a class as profoundly amazing, intricate or basic software programs.

IV. Proposed Strategy Vector Machine Tree Classification

We have proposed a vector machine tree classification model for the software fault forecast model. This proposed model will help to increase the positive accuracy and efficiency of the software fault forecast model. The proposed method called TSMO or TSVM, which based on binary tree classification. This gives an improvement in the capable computation of multiple trees and increases the classification accuracy of the vector machine. This proposed TSMO/TSVM has a structure of binary decision trees, which depends on a sequential minimal optimization decision tree. An origin is the starting point of this procedure where minimal optimization selected binary branches from one tree.

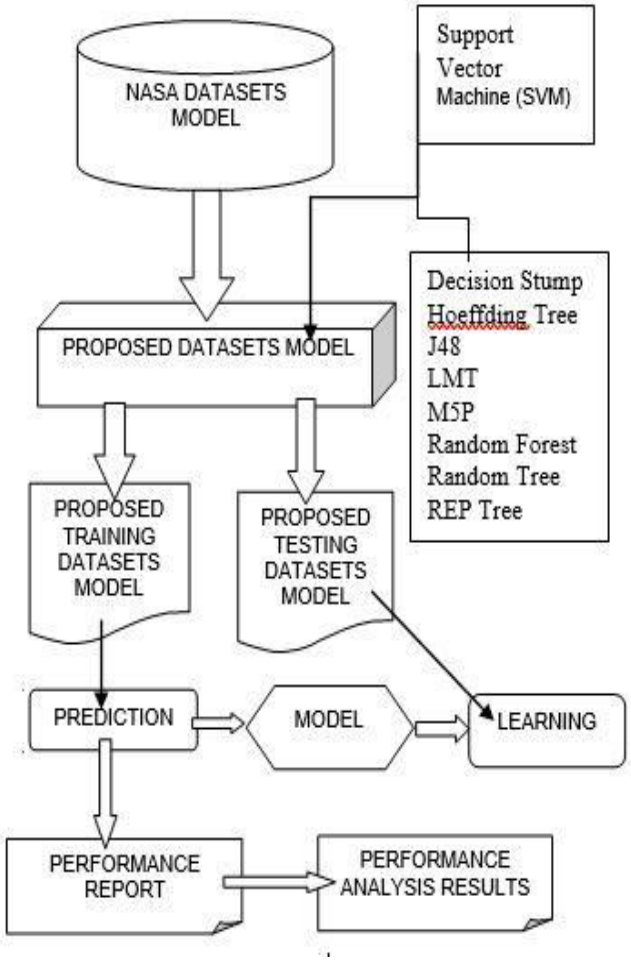
This process frequent performs until the final leaf of the tree becomes visible when the final leaf appears which usually characterized the classification of three. We have used multiple trees for getting more accuracy and efficiency.

V. Research Proposed Tree Vector Machine Model

This is our proposed method flow chart for our research where we have used multiple tree classification with the help of a vector machine called TSMO/TSVM. We have taken NASA MDP datasets in which data is classified into the defective models and non-defective models. Our datasets models are further defined in table1. We have used WEKA 3.9 to analysis the efficiency and positive accuracy of the datasets model. The analysis measure unit is the area under cure, f-measure positive accuracy, TP-Rate and correctly classifies instances accuracy.

TABLE I. NASA Datasets Model

S.NO	Dataset	Attribute	Model	Defective	Non Defective
1	MC2	22	522	107	415
2	AR4	30	107	20	87
3	PC5	39	17186	516	16670
4	MC1	39	1988	46	1942
5	PC1	38	705	61	644
6	AR5	30	36	8	28
7	KC3	40	194	36	158
8	JM1	22	7782	1672	6110
9	KC2	22	522	107	415
10	AR1	30	121	9	112
11	AR6	30	101	115	86
12	MW1	38	253	27	226
13	AR3	30	63	8	55
14	CM1	38	327	42	285



Flow Chart 2: Research Propose Model

Volume: 12 Issue:04 Pages: 4650-4655(2021) ISSN: 0975-0290

VI. EXPERIMENTS & RESULTS

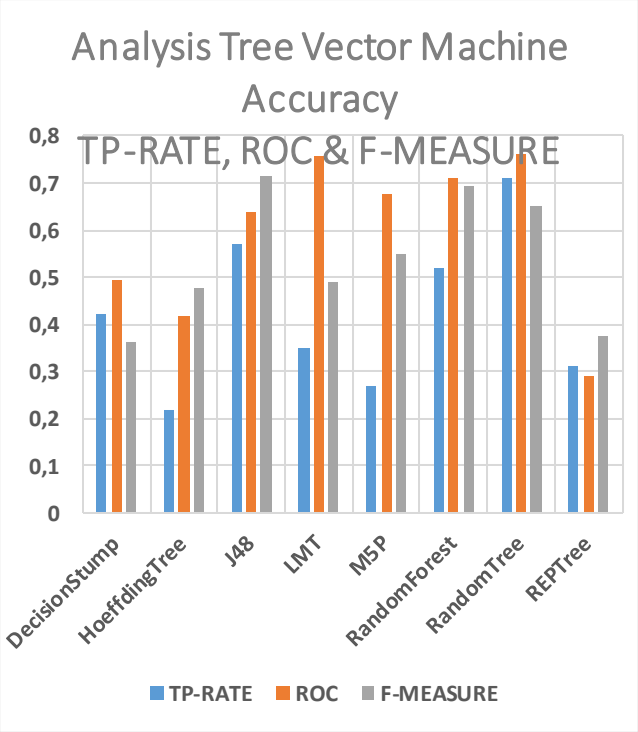


Fig. 1. Positive Accuracy TP-RATE, F-MEASURE & ROC Curve

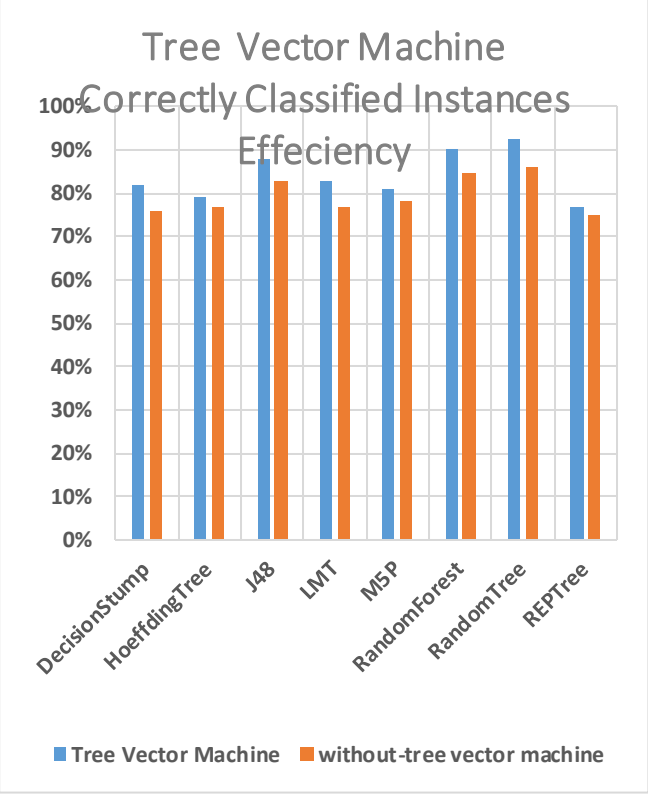


Fig. 2. Comparatively Analysis b/w tree vector & without-tree vector.

TABLE II. COMPARATIVELY ANALYSIS IMPROVEMENT

Name of Classifier	Tree Vector Machine	Without Tree Vector Machine	Improvement
Decision Stump	82%	76%	6%
Hoeffding Tree	79%	77%	2%
J48	88%	83%	5%
LMT	83%	77%	6%
M5P	81%	78%	3%
Random Forest	90%	84.5%	5.5%
Random Tree	92.5%	86%	6.2%
REP Tree	77%	75%	2%

During the experiments of the proposed model, the results are illustrated in fig 1& 2. In fig1, multiple tree classification has performed here with a vector machine model, where we have analyzed that J48, random forest and random tree have a very high positive accuracy F-Measure rate as compare to other trees. But the performance of REP Tree, Hoeffding Tree, and Decision Stump is not so good at all measure rates. The experiment's analysis results showed that not every tree classifier could be good in all in measure unit. Our analysis depends on the accuracy of tree classification that how much accuracy has increased in which tree classifier. In fig2, the efficiency of every classifier is measured in terms of correctly classified instances rate, where an experiment showed that the efficiency of overall classified is increased with the proposed model as compare to without using any method. The efficiency of j48, random forest and the random tree is also in high-rate performance rather than others.

VII. CONCLUSION

In our research paper, we have analyzed the software fault forecast model, we have used 14 NASA MDP datasets models. These models are defected and non-defected datasets models. We have proposed a tree vector machine model called TSMO/TSVM, where we have used multiple tree classifiers with the help of the vector machine. We have used WEKA 3.9 for analysis of the efficiency and positive accuracy of the datasets model. During the analysis of the experiment, j48, random forest and random tree have increased their performance in accuracy as well as efficiency. However, the performance of REP tree, Hoeffding tree and Decision stump is not so good at all measure rates.

REFERENCES

[1] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning,20(3), 273-297.

[2] Dietterich, T. G., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. arXiv preprint cs/9501101.

[3] K.O. Elish, M.O. Elish, Predicting defect-prone software modules using support vector machines, Journal of Systems and Software 81 (2008) 649–660.

[4] L. Etzkorn, J. Bansiya, C. Davis, Design and code complexity metrics for OO classes, Journal of Object-Oriented Programming 12 (1999) 35–40.

[5] N. Fenton, S. Pfleeger, Software Metrics: A Rigorous and Practical Approach, vol. 5, PWS Publishing Company (An International Thomson Publishing Company), 1997.

[6] F. Garcia, M. Bertoa, C. Calero, A. Vallecillo, F. Ruiz, M. Piattini, M. Genero,Towards a consistent terminology for software measurement, Information and Software Technology 48 (2006) 631–644.

[7] Fenton, N. E., Marsh, W., Neil, M., Cates, P., Forey, S. and Tailor, T. Making Resource Decisions for Software Projects. In Proceedings of 26th International Conference on Software Engineering (ICSE 2004), (Edinburgh, United Kingdom, May 2004) IEEE Computer Society 2004, ISBN 0- 7695-2163-0, 397-406

[8] Fenton, N. E. and Neil, M. A Critique of Software Defect Prediction Models, IEEE Transactions on Software Engineering, 25(5), 675-689, 1999.

[9] Alenezi, , et al, "Efficient bug triaging using text mining. Journal of Software "8;2003. no. 9.

[10] Runeson, , et al, "Detection of duplicate defect reports using natural language processing", In Software Engineering, 2007. ICSE 2007. 29th International Conference on., IEEE ; 2007.pp. 499-510.

[11] S. Adiu , et al , "Classification of defects in software using decision tree algorithm", International Journal of Engineering Science and Technology (IJEST), Vol. 5, Issue 6, pp. 1332-1340. 12.

[12] Murphy, Gail C., and D. Cubranic. Automatic bug triage using text categorization. In Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering; 2004.

[13] Anvik, et al, Who should fix this bug?. InProceedings of the 28th international conference on Software engineering,. ACM ;2006. pp. 361-370 .

[14] Cathrin et al, 2007, "How Long will it Take to Fix This Bug? Intl. conf. on software engineering", IEEE Computer Society Wanshington, DC, USA, pp. 1-8.

[15] Cathrin et al, "Predicting Effort to Fix Software Bugs", Proceedings of the 9th Workshop Software Reengineering,2007.

[16] Sunghun Kim, Kai Pan, E. James Whitehead, Jr., 2006, Memories of Bug Fixes, SIGSOFT'06/FSE-14, November 5–11, Portland, Oregon, USA.

[17] Menzies, et al, "Automated severity assessment of software defect reports in Software Maintenance", 2008. ICSM 2008. IEEE International Conference on, IEEE; 2008.pp. 346-355.

[18] T. Gyim´othy, R. Ferenc, and I. Siket. Empirical validation of objectoriented metrics on open source software for fault prediction. IEEE Transactions on Software Engineering (TSE), 31(10):897–910, 2005.

[19] M. A. Hall. Correlation-based feature selection for machine learning.1999.

[20] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell. A systematic literature review on fault prediction performance in software engineering. IEEE Transactions on Software Engineering, 38(6):1276–1304, 2012.

[21] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. Radiology, 143(1):29–36, 1982.

[22] A. E. Hassan. Predicting faults using the complexity of code changes. In ICSE, pages 78–88, Vancouver, Canada, 2009. IEEE Press.

[23] H. Hata, O. Mizuno, and T. Kikuno. Bug prediction based on finegrained module histories. In Proceedings of the 34th International Conference on Software Engineering, pages 200–210. IEEE Press, 2012.

[24] C. Huang, L. Davis, and J. Townshend. An assessment of support vector machines for land cover classification. International Journal of remote sensing, 23(4):725–749, 2002.

[25] Q. Huang, X. Xia, and D. Lo. Supervised vs unsupervised models: A holistic look at effort-aware just-in-time defect prediction. In Software Maintenance and Evolution (ICSME), 2017 IEEE International Conference on, pages 159–170. IEEE, 2017.

[26] Z. Jiang and S. Sarkar. Free software offer and software diffusion: The monopolist case. ICIS 2003 proceedings, page 81, 2003.

[27] Y. Kamei, S. Matsumoto, A. Monden, K.-i. Matsumoto, B. Adams, and A. E. Hassan. Revisiting common bug prediction findings using effort-aware models. In Software Maintenance (ICSM), 2010 IEEE International Conference on, pages 1–10. IEEE, 2010.

[28] Y. Kamei, E. Shihab, B. Adams, A. E. Hassan, A. Mockus, A. Sinha, and N. Ubayashi. A large-scale empirical study of just-in-time quality assurance. IEEE Transactions on Software Engineering, 39(6):757–773, 2013.

[29] W. M. Khaled El Emam and J. C. Machado. The prediction of faulty classes using object-oriented design metrics. Journal of Systems and Software, 56(1):63–75, 2001.

[30] F. Khomh, M. Di Penta, Y.-G. Gu´eh´eneuc, and G. Antoniol. An exploratory study of the impact of antipatterns on class change-and faultproneness. Empirical Software Engineering, 17(3):243–275, 2012.

[31] S. Kim, E. J. Whitehead Jr, and Y. Zhang. Classifying software changes: Clean or buggy? IEEE Transactions on Software Engineering, 34(2):181–196, 2008.

[32] J. Kittler et al. Pattern recognition. a statistical approach. 1982.

[33] S. Kpodjedo, F. Ricca, P. Galinier, Y.-G. Gu´eh´eneuc, and G. Antoniol. Design evolution metrics for defect prediction in object oriented systems. Empirical Software Engineering, 16(1):141–175, 2011.

[34] K. Krippendorff. Content analysis: An introduction to its methodology. Sage, 2004.

[35] M. Lanza, A. Mocci, and L. Ponzanelli. The tragedy of defect prediction, prince of empirical software engineering research. IEEE Software, 33(6):102–105, 2016.

[36] M. M. Lehman and L. A. Belady. Program evolution: processes of software change. Academic Press Professional, Inc., 1985.

[37] C. Lewis, Z. Lin, C. Sadowski, X. Zhu, R. Ou, and E. J. Whitehead Jr. Does bug prediction support human developers? Findings from a Google case study. In Proceedings of the 2013 International Conference on Software Engineering, ICSE 2013, pages 372–381. IEEE Press, 2013.

[38] Fenton, N. E. and Neil, M. SCULLY: Scaling up Bayesian Nets for Software Risk Assessment, Queen Mary University of London, www.dcs.qmul.ac.uk/research/radar/Projects, 2001.

[39] Fenton, N. E. and Pfleeger, S.L. Software Metrics: A Rigorous and Practical Approach (2nd Edition), PWS, ISBN: 0534-95429-1, 1998.

[40] Jensen, F.V. An Introduction to Bayesian Networks, UCL Press, 1996.

[41] Jones, C. Programmer Productivity, McGraw Hill, 1986.

[42] Jones, C. Software sizing, IEE Review 45(4), 165-167, 1999.

[43] Koller, D., Lerner, U. and Angelov, D. A General Algorithm for Approximate Inference and its Application to Hybrid Bayes Nets, In Proceedings of the 15th Annual Conference on Uncertainty in AI (UAI), Stockholm, Sweden, August 1999, pages 324—333

[44] Kozlov, A.V. and Koller, D. Nonuniform dynamic discretization in hybrid networks, Proceedings of the 13th Annual Conference on Uncertainty in AI (UAI), Providence, Rhode Island, August 1997, pages 314--325.

[45] I. Gondra, Applying machine learning to software fault-proneness prediction, Journal of Systems and Software 81 (2008) 186–195.

[46] L.A. Goodman, Snowball sampling, The Annals of Mathematical Statistics 32 (1961) 148–170.

[47] G.a. Hall, J.C. Munson, Software evolution: code delta and code churn, Journal of Systems and Software 54 (2000) 111–118.

[48] T. Hall, S. Beecham, D. Bowes, D. Gray, S. Counsell, A systematic literature review on fault prediction performance in software engineering, IEEE Transactions on Software Engineering (2011) 1–31.

[49] M.H. Halstead, Elements of Software Science, Elsevier Science Inc., New York, NY, USA, 1977.

[50] B. Henderson-Sellers, Software Metrics, Prentice-Hall, Hemel Hempstaed, UK, 1996.

[51] M. Hitz, B. Montazeri, Measuring coupling and cohesion in object-oriented systems, in: Proceedings of the International Symposium on Applied Corporate Computing, vol. 50, 1995, pp. 75–76.

[52] J.K. Methew ISO/IEC, IEEE, ISO/IEC 12207:2008 systems and software engineering software life cycle processes, 2008.

[53] J.M. Juran, F.M. Gryna, Juran's Quality Control Handbook, McGraw-Hill, 1988.

[54] S. Kanmani, V.R. Uthariaraj, V. Sankaranarayanan, P. Thambidurai, Objectoriented software fault prediction using neural networks, Information and Software Technology 49 (2007) 483–492.

[55] T.M. Khoshgoftaar, N. Seliya, Comparative assessment of software quality classification techniques: an empirical case study, Empirical Software Engineering 9 (2004) 229–257.

[56] Gnanambal, Thangaraj. Classification algorithm with attribute selction: an evaluation study WEKA. Int. J. Advanced Networking and Applications Volume: 09 Issue: 06 Pages: 3640-3644 (2018) ISSN: 0975-0290.

[57] Raghavendra B. K., Dr. Jay B. Simha. Evaluation of Feature Selection Methods for Predictive Modeling Using Neural Networks in Credits Scoring. Int. J. Advanced Networking and Applications Volume:02, Issue: 03, Pages: 714-718 (2010).