International Journal of
Technology and Systems
(IJTS)

DEVELOPING CONCEPTUAL FRAMEWORK OF SOFTWARE DEFECT PREDICTION IN SOFTWARE TESTING: THE CASE OF ETHIOPIAN SOFTWARE INDUSTRIES

Musa Dima Genemo

International Journal of Technology and Systems
ISSN 2518-881X (Online)
Vol.8, Issue 1, No.1, pp 1 – 13, 2023

www.iprjb.org

Developing Conceptual Framework of Software Defect Prediction in Software Testing: The Case of Ethiopian Software Industries

[1*]Musa Dima Genemo (2013)
Department of Computer Engineering
[*]Corresponding Author's Email:
musa.ju2002@gmail.com

Article History

Received 15th February 2023

Received in Revised Form 25th February 2023

Accepted 8th March 2023

Abstract

Purpose: Software defect prediction is one of the most active research areas in software engineering. As our dependency on software is increasing, software quality is becoming gradually more and more important in present era. Software used almost everywhere and in every tread of life. Software consequences such as fault and failures may diminish the quality of software which leads to customer dissatisfaction. Different points that are related to the current work had been discussed.

Methodology: This research contains a detailed explanation of the scientific methods and methodologies used for the design of proposed framework. Primary and secondary source of data has been used.

Findings: Due to the tremendous amount of data generated daily from fields such as business that software has resulted in the generation of tremendous amount of defected data. As the organizations struggle to handle and utilize effectively all the information available in order to provide better products and services and gain competitive advantage, defect separation and the so called "big data" analytics are two fields that currently constitute matter of concern and discussion.

Unique Contribution to Theory, Practice and Policy: There are different research areas in defect prediction using big data analytics defect segmentations one of the issues. Defect Segmentation is an important component of much organization, and the algorithm designed in this work is expected to be a significant contribution to the field, and mainly to researchers working on various aspects of defect prediction defect segmentation using big data analytics. Therefore, the researchers in the area can use the algorithm or the implemented system for processing segmentation as component in their research, mainly on machine learning applications.

Keywords: *Software Defect, Defect Prediction, Software Testing, Software Security*

INTRODUCTION

Software defect prediction is one of the most active research areas in software engineering. As our dependency on software is increasing, software quality is becoming gradually more and more important in present era. Software used almost everywhere and in every tread of life. Software consequences such as fault and failures may diminish the quality of software which leads to customer dissatisfaction.

Software life cycle is a human activity, so it is impossible to prevent the injection of defects but it is possible to produce the software with few defects. (Naheed Azeem, Shazia Usmani, 2011) To deliver defect free software it is imperative to predict and fix the defects as many as possible before the product delivers to the customer. Due to the increasing of complexity and the constraints under which the software is developed, it is too difficult to produce quality software. Therefore, defect prediction before delivery can contribute significantly to the success of project in terms of quality and cost. Defects in software product cause much loss of time and money.

Learning from past experience, it would be possible to predict defects in advance for new software products. Finding and fixing the defects after delivery usually consumes a large portion of the project budget. The aim of this research is to explore the different issues and problems in the area of defect prediction as well as provide the solutions to improve the product quality via defect prediction mechanism (20).

Researchers have devised and implemented excess of bug prediction approaches varying in terms of accuracy, complexity and the input data they require. However, the absence of an established benchmark makes it hard, if not impossible, to compare approaches. In this paper benchmark for defect prediction has been resented, in the form of a publicly available data set consisting of several software systems and provides an extensive comparison of the explanative and predictive power of well-known bug prediction approaches. Recent studies of software defect prediction typically produce datasets, methods and frameworks which allow software engineers to focus on development activities in terms of defect-prone code, thereby improving software quality and making better use of resources.

Many software defect prediction datasets, methods and frameworks are published disparate and complex, thus a comprehensive picture of the current state of defect prediction research that exists is missing. This study aims to identify and analyze the research trends, datasets, methods, and frameworks used in software defect prediction. And finally, the study had contributed a framework for defect prediction. This study has been undertaken as a systematic literature review as a process of identifying, assessing, and interpreting all available research evidence with the purpose to provide answers for specific research questions. Analysis of the selected primary studies revealed that current software defect prediction research focuses on five topics and trends: estimation, association, classification, and clustering and dataset analysis. The total distribution of defect prediction methods is as follows, 77.46% of the research studies are related to classification methods, 14.08% of the studies focused on estimation methods, and 1.41% of the studies concerned on clustering and association methods. In addition, 64.79% of the research studies used

public datasets and 35.21% of the research studies used private datasets. Nineteen different methods have been applied to predict software defects. From the nineteen methods, seven most applied methods in software defect prediction are identified [3].

Researchers proposed some techniques for improving the accuracy of machine learning classifier for software defect prediction by assembling some machine learning methods, by using boosting algorithm, by adding feature selection and by using parameter optimization for some classifiers. This study has also focused on identifying the frameworks that are highly cited and filling gaps of the existing framework.

Statement of the Problem

It is known that any software and any service delivered for customer is measured by security, energy efficiency, reliability, error free, and fault tolerance level. Having this in mind if we examine the current defect prediction method in software testing in terms of defect prediction level currently Users, company and developers may face many problems in different countries.

The current Defect prediction mechanism has no benchmark to predict software as defect and defect free. However, the absence of an established benchmark makes it hard, it is not impossible, to compare approaches. The absence of predicting whether a given code segment is defected or not and predicting the magnitude of the possible defect, if any, with respect to various viewpoints such as density, severity, or priority. This may open the door for the system familiarity reason.

In Addition to the above listed problem there is also a gap in differing version control and change management systems in testing, Inconsistent identification of software defects in defect prediction, Lack of mapping between software structures (files) and requirements are the issues in defect prediction.

The above-mentioned problem is the most critical from point of view of the research of this paper and that must be solved by this study. The current available framework and study focus on functional aspect of the system based on generating data from functional requirement of the system. Even in this case there are many gaps in considering boundary testing. Many literatures reviewed for purpose of this study focus on what function and what security the system should have. The other gap expected to be filled in this study is estimating the defect causing, potential of a given software project has a very critical value for the reliability of the project.

The major purpose of this study is to investigate framework of software defect prediction in software testing and implementing a framework defect prediction.

Organization of the Thesis

The manuscript is written according to following sequence. Sections 1 and 2 are comprised of the introduction part and the literature insight respectively. Section 3 encompasses the proposed approach. Section 4 depicts results and discussion with details of performance evaluation experiments and Time graph. Section 5 shows the paper's conclusions

LITERATURE REVIEW

This chapter present review of literature and related work of software defect prediction. Different points that are related to the current work had been discussed. Extensive literature review is conducted on software defect prediction in order to obtain in-depth understanding of the area and to find the best conceptual for software defect prediction.

The history of defect prediction studies was about last 50 years or 5 decades. The first study estimating the number of defects was conducted by (Akiyama., 1971). Based on the assumption that complex source code could cause defects, Akiyama built a simple model using lines of code (LOC) since LOC might represent the complexity of software systems. However, LOC is too simple metric to show the complexity of systems. In this reason, (McCabe, 1976) and (Halstead, 1977) proposed the cyclamate complexity metric and Halstead complexity metrics in respectively. These metrics were very popular to build models for estimating defects in 1970s and the early of (N. Fenton and M. Neil, 1999).Having said that though, the models studied in that period were not actually prediction model but just fitting model that investigated the correlation between metrics and the number of defects .These models were not validated on new software modules. To resolve this limitation of previous studies (Shen *et a*l, 1985) built a linear regression model and test the model on the new program modules.

However, (Munson *et al* , 1992)claimed that the state-of-the art regression techniques at that time were not precise and proposed classification models that classify modules into two groups, high risk and low risk. The classification model actually achieved 92% of accuracy on their subject system. However, Munson et al.'s study still have several limitations such as no metrics for object-oriented (OO) systems and few resources to extract development process data. As Shen et al. pointed out at that time, it was not possible to collect error fix information informally conducted by individual developers in unit testing phase's .In terms of OO systems, (Chidamber) and (Kemerer) proposed several object-oriented metrics in 1994 and was used by (Basili et al) to predict defects in object-oriented system.

In 1990s, version control systems were getting popular, development history was accumulated into software repositories so that various process metrics were proposed from the middle of 2000. In 2000, there had been existed several limitations for defect prediction. The first limitation was the prediction model could be usable before the product release for the purpose of quality assurance. However, it would be more helpful if we can predict defects whenever we change the source code. To make this possible (Mockus *et al*).Proposed a defect prediction model for changes. Recently, this kind of models is called as just-in-time (JIT) defect prediction models. JIT prediction models have been studied by other researchers in recent years.

Software defects are errors, flaws, bugs, mistakes, failures or faults in computer programs or systems that generate inaccurate/unexpected outcome, or preclude software from its intended behavior. (Gayathri M, A. Sudha, 2014) Software defect prediction is a current and hot issue of software companies. Due to many reasons software's are always failing. The reason of this faulty is due to penetrating defect software to the market without appropriate testing. To solve this

problem many studies has been employed. But not yet solved the problem fully from the beginning. According software defect prediction definition software defect prediction is testing software for error and prone free.

Inline to this (Turhan, 2007)has proposed a statistical defect predictor model with two major differences from the existing ones. Using only static code measures for the research avoids any kind of human error and subjectivity from the datasets. As noted by(Wahyudin, Ramler, and Biffl, 2009)a framework proposed for conducting software defect prediction as an aid for the practitioner establishing defect prediction in the context of a particular project or organization and as a guide to the body of existing studies on defect prediction.

As to (Umar, 2013)explained statistical model, defect prediction for upcoming software releases or projects. To predict software testing defects using statistical models and evaluate the accuracy of the statistical defect prediction model, he used 20 past release data points of software project, parameters and builds a model by applying descriptive statistics, correlation and multiple linear regression models with 95%confidence intervals (CI). They analyzed an extensive historical dataset of software releases to identify factors influencing parameters of defect prediction model. He found strong correlation between defects and test team size, total number of test cases executed, total number of components delivered. In this analysis he used multiple regression analysis for estimating software defects. Inline to this (Xiaoxing , Ke , and Xin , 2014)introduced a learning-to-rank approach to construct software defect prediction models by directly optimizing the ranking performance. The work includes two aspects: one is a novel application of the learning-to-rank approach to real-world data sets for software defect prediction, and the other is a comprehensive evaluation and comparison of the learning-to-rank method against other algorithms that have been used for predicting the order of software modules according to the predicted number of defects.

In addition to this (Saiqa , Luiz and Faheem , 2015)Study indicates that the public available data sets of software modules and provides comparative performance analysis of different machine learning techniques for software bug prediction. Results showed most of the machine learning methods performed well on software bug datasets. According to (Saiqa , Luiz and Faheem , 2015)Effective bug's prediction is totally dependent on a good prediction model. As this study indication the software defect can be predicted if and only if the benchmark of defect is clearly stated before starting software testing.

As to (Wanjiang ,Lixin ,Tianbo,Xiaoyan, Yi , 2014)an approach to predict residual defects, which applies machine learning algorithms (classifiers) and defect distribution model includes two steps. Firstly, use machine learning Algorithms to get defect classification table, then confirm the defect distribution trend referring to several distribution models. According to the study finding, before starting software testing, we have to have benchmark data set that helps as to classify software as defect or not defect.

Inline to this (Bharavi and K.K., 2012)proposed a new Support Vector based Fuzzy Classification System (SVFCS) for defective module prediction. In the proposed model an initial rule set is constructed using support vectors and Fuzzy logic. Rule set optimization is done using Genetic

algorithm. The new method has been compared against two other models reported in recent literature viz. Naive Bayes and Support Vector Machine by using several measures, precision and probability of detection and it is found that the prediction performance of SVFCS approach is generally better than other prediction approaches. From this study there is doubt from researchers that the classification based on support vector machine and fuzzy logic. Hence testing without benchmark data resulted software faulty. According to (M .and G.S. Anandha, 2015)carried out Experiments on analyzing the defect prediction using different types of classifiers such as NB, SVM, and KNN etc. The classification accuracy of the SVM classifier performs better when compared to other classifiers. The advantage of SVM is that they provide better performance. The main disadvantage of SVM is that it does not work well on public datasets.

(Kritika Verma Pradeep Kumar , 2015)Researchers have performed descriptive survey and analysis to provide efficient results for defect prediction in software systems and various techniques have been used in order to arrange their performance capacity. Neural Network being the best followed by Decision Tree and Bayesian Network, then the SVM and lastly comes the KNN method. It is suggested that Ensemble Machine learning and One class SVM are two areas that can be used extensively in future. But according to these researchers SVM technique does not perform well so it's better to focus on Ensemble Machine Learning in future to predict defects. (Fenton,Norman E.,Krause,Paul.,Neil,Martin., 2001)The application of neural networks to the problem of defect prediction has received a great of attention. Neural network has successfully been applied to predict defects in a chemical processing plant .the results were 10 to 20 times better than the application of traditional method.

(Song,Q., Jia,Z.,Shepperd,M.,Ying,S.,& Liu,J, 2011)Suggested a general software defect prediction framework supporting unbiased/comprehensive comparison between competing prediction systems. The framework includes scheme evaluation and defect prediction. Scheme evaluation analyzes prediction performance of competing schemes for specific historical data sets. The defect predictor constructs models based on evaluated learning schemes predicting software defects with new data according to a constructed model. To demonstrate the proposed framework's performance, simulations were undertaken on publicly available software defect datasets. Results demonstrated the requirement of various learning schemes for differing datasets (i.e., no scheme dominates) and that small details in conducting evaluations conduct completely reverses findings. The proposed framework is effective and not liable for bias than earlier approaches.

(Juan and Marcelo , 2015)The researchers have included framework more combinations of learning schemes than other proposals. There are more possibilities to find better learning schemes for each data set. The genetic approach has presented better performance in the majority of the cases, representing eight of ten data sets. The researcher put the gap as it is necessary to include more data sets with different size, noise level and imbalance data from public and private repositories.

(K.B.S , Dr.B.V., Dr K.V, 2013)Describes framework to produce software defect from the historical database and also present one pass data mining algorithm used find rules to predict

software defects. The experimental results show that, one pass algorithm generate rules for software defect prediction with consider amount of time and with better performance.

According to (Juan Marcelo and K.B.S , 2013)Studies in Software defect prediction framework for example, 2013 (Dr K.V.Sambasiva and Marcelo , 2015). Therefore this research gap initiates me to carry out research in this area with aim to asses' implementation of developing conceptual framework of software defect prediction .This study therefore focus on the gap between what developing conceptual framework for software defect prediction in software testing in the case of Ethiopian software industries under take theoretically and what it in fact (implementation) achieves in context of Ethiopian software company.

Research Contribution

This research contains a detailed explanation of the scientific methods and methodologies used for the design of proposed framework. The constructionist of the framework was based on a combination of literature and empirical data collected from the users and professional persons (Software developer).

Table 1: Research placed into DRM (Design Research Methodology)

Identification	**Problem definition**
Objective for solution	Review on defect prediction Review on software testing
Design and development	Defect prediction goal Discussion with professionals
Demonstration	Showing prototype to advisor Discussion with Professionals
Evaluation	Expert evaluation (software developer)
Communication	Suggestion from expert Evaluation Future work recommendations

Data Source

There is data source for this study. Primary and secondary source of data has been used. The primary data source of this study is software developer of selected software company and selected software development unit of the university. The secondary data source of this study is literature review, software company report, professional blogs and forums. Testing is the traditional process for identifying defects. However, when projects' size grows in lines of code and complexity algorithm finding and fixing errors gets more difficult and computationally expensive with the use of sophisticated testing and evaluation procedures. According to Boehm study result finding and fixing a problem after delivery is more expensive, in terms of cost and effort, than fixing it during the early stages of software life cycle[]. Early detection of fault-prone software components enables verification experts to concentrate their time and resources on the problem areas of the software system under development.

Software Defect Prediction Metrics

Defect prediction metrics play the most important role to build a statistical prediction model. Most defect prediction metrics can be categorized into two kinds: code metrics and process metrics. Code metrics are directly collected existing source code while process metrics are collected from historical information archived in various software repositories such as version control and issue tracking systems. Code metrics also known as product metrics measure complexity of source code. Its ground assumption is that complexity source is more bug prone.

Defect Prediction Framework

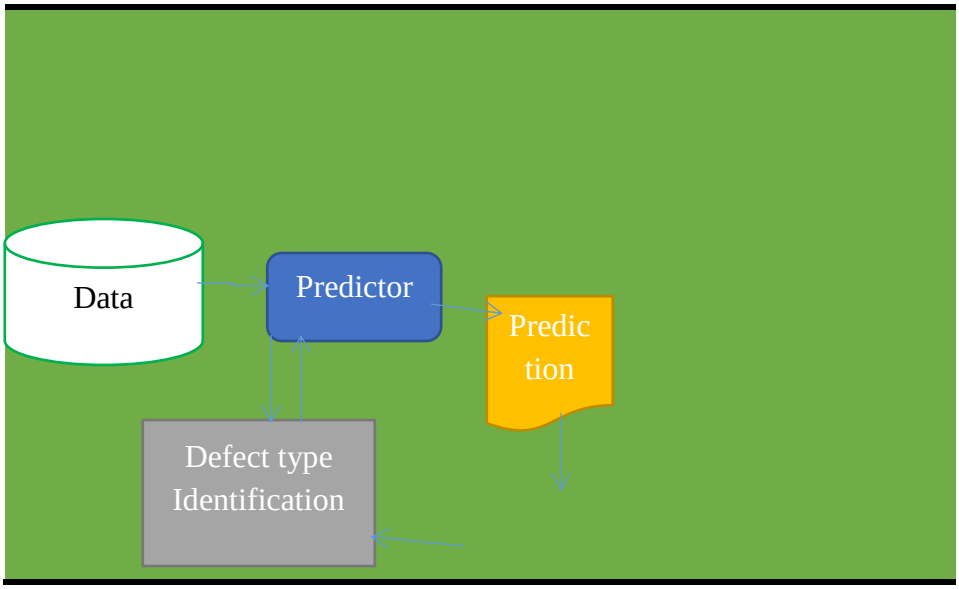

Figure 1: Defect Prediction Framework

Table 2: Data Adopted from NASA Metrics Data Program (MDP)

Data Set	MDP		
CMI	505		
JM1	10 878		
KC1	2107		
KC2	n.a		
KC3	458		
KC4	125		
MC1	9466		
MC2	161		
MW1	403		
PC1	1107		
PC2	5589		
PC3	1563		
PC4	1458		
PC5	17186		

The most common usage for the NASA data sets (as reported in the literature) is in binary classification experiments. Typically, a classifier is trained on binary labeled data, and then each new set of module metrics is predicted as belonging to either a 'faulty' module, or a 'non-faulty' module. This is clearly a huge simplification of the real world, for two main reasons. Firstly, fault quantity is disregarded: there is typically no distinction between a module with one reported fault and a module with 31 reported faults, they are both simply labeled as 'faulty'. Secondly, fault severity is disregarded: there is typically no distinction between a trivial fault and a life-threatening fault. Despite these crude simplifications, binary classification defect prediction studies continue to be very prolific. It is widely accepted by the data mining community that in order to accurately assess the potential real-world performance of a classification model, the model must be tested against entirely different data from that upon which it was trained.

FINDINGS

Due to the tremendous amount of data generated daily from fields such as business that software has resulted in the generation of tremendous amount of defected data. As the organizations struggle to handle and utilize effectively all the information available in order to provide better products and services and gain competitive advantage, defect separation and the so called "big data" analytics are two fields that currently constitute matter of concern and discussion.

During the research, a comprehensive literature review in the fields of defect prediction, user's engagement, big data and data mining was conducted. The main objective of the research was to answer the main research question. The main deliverable of this research, which constitutes an answer to the main research question, was a framework that shows defect prediction types able to assist error free and user friendly software objectives, and the techniques that can be used for big data analysis are suitable for segmenting e-commerce users according to each of, the e-commerce user's segmentation types

CONCLUSION AND RECOMMENADTIONS

Conclusion

Defect segmentation using big data analytics is a task of defect prediction into homogenous group based on their preference. It is a research area in the field of software engineering in relation with big data analytics. The problem to segment defect using big data analytics is software businesses are currently building their defect testing strategies and thus they do not have well defined framework. Therefore, marketers do not provide data analysts with the appropriate information, while proceeding with valuable and effective defect segmentation becomes difficult, therefore while segmenting we have no one fit-size segmentation. Defect prediction is a core process for assisting a software marketing strategy. However, there is limited scientific research related to the field. Huge amount of defect prediction data is continuously generated. However, there was no scientific research found for the use of big data tools in defect prediction. In the world of business, a gap between defect prediction and data analysts emerges. Normally, software sellers should be able to select combinations.

Recommendations

There are different research areas in defect prediction using big data analytics defect segmentations one of the issues. Defect Segmentation is an important component of much organization, and the algorithm designed in this work is expected to be a significant contribution to the field, and mainly to researchers working on various aspects of defect prediction defect segmentation using big data analytics. Therefore, the researchers in the area can use the algorithm or the implemented system for processing segmentation as component in their research, mainly on machine learning applications. As a future work Researcher would like suggest the following points: A general guideline for effective defect prediction segmentation is needed, while the opportunities that machine learning offer for defect prediction segmentation should be further explored. In feature research, the frameworks could be tested as a whole on more than one real situation. The actual usefulness of defect prediction types for each of the objectives, according to the first framework can be tested on a real situation. Starting with a certain defect objective certain prediction segmentation types can be selected to be analyzed for creating actionable defect free SW.

REFERENCES

D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. Machine Learning, 6(1):1573–0565, January 1991.

M. L. Berenson, M. Goldstein, and D. Levine. Intermediate Statistical Methods and Applications: A Computer Package Approach. Prentice-Hall, Englewood Cliffs, NJ, 1983.

G. Boetticher, T. Menzies, and T. Ostrand. Promise repository of empirical software engineering data. http://promisedata.org/, 2007.

A. C. Cameron and P. K. Trivedi. Regression Analysis of Count Data. Cambridge University Press, 1998.

Z. Chen, T. Menzies, D. Port, and B. Boehm. Finding the right data for software cost modeling. IEEE Software, 22(6):38–46, November 2005.

J. T. de Souza, N. Japkowicz, and S. Matwin. Stochfs: A framework for combining feature selection outcomes through a stochastic process. In Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, pages 667–674, Porto, Portugal, October 3-7 2005.

T. Fawcett. An introduction to ROC analysis. Pattern Recognition Letters, 27(8):861–874, June 2006.

U. M. Fayyad and K. B. Irani. On the handling of continuous-valued attributes in decision tree generation. Machine Learning, 8:87–102, 1992.

K. Gao, T. M. Khoshgoftaar, and H. Wang. An empirical investigation of filter attribute selection techniques for software quality classification. In Proceedings of the 10th IEEE International Conference on Information Reuse and Integration, pages 272–277, Las Vegas, Nevada, August 10-12 2009.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. Journal of Machine Learning Research, 3:1157–1182, March 2003.

M. A. Hall and G. Holmes. Benchmarking attribute selection techniques for discrete class data mining. IEEE Transactions on Knowledge and Data Engineering, 15(6):1437 – 1447, Nov/Dec 2003.

S. Haykin. Neural Networks: A Comprehensive Foundation. Prentice-Hall, 2nd edition, 1998.

G. Ilczuk, R. Mlynarski, W. Kargul, and A. Wakulicz-Deja. New feature selection methods for qualification of the patients for cardiac pacemaker implantation. In Computers in Cardiology, 2007, pages 423–426, Durham, NC, USA, 2007.

Y. Jiang, J. Lin, B. Cukic, and T. Menzies. Variance analysis in software fault prediction models. In Proceedings of the 20th IEEE International Symposium on Software Reliability Engineering, pages 99–108, Bangalore-Mysore, India, Nov. 16-19 2009.

G. H. John and P. Langley. Estimating continuous distributions in Bayesian classifiers. In Proceedings of Eleventh Conference on Uncertainty in Artificial Intelligence, volume 2, pages 338–345, San Mateo, 1995.

K. Jong, E. Marchiori, M. Sebag, and A. van der Vaart. Feature selection in proteomic pattern data with support vector machines. In Proceedings of the 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, Oct 7-8 2004.

S. A. Julious. Using confidence intervals around individual means to assess statistical significance between two means. Pharmaceutical Statistics, 3:217–222, 2004.

T. M. Khoshgoftaar, L. A. Bullard, and K. Gao. Attribute selection using rough sets in software quality classification. International Journal of Reliability, Quality and Safety Engineering, 16(1):73–89, 2009.

T. M. Khoshgoftaar, M. Golawala, and J. V. Hulse. An empirical study of learning from imbalanced data using random forest. In Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence, volume 2, pages 310–317, Patras, Greece, Oct. 29-31 2007.

T. M. Khoshgoftaar, Y. Xiao, and K. Gao. Assessment of a multi-strategy classifier for an embedded software system. In Proceedings of 18th IEEE International Conference on Tools with Artificial Intelligence, pages 651–658, Washington, DC, USA, November 13-15 2006.

K. Kira and L. A. Rendell. A practical approach to feature selection. In Proceedings of 9th International Workshop on Machine Learning, pages 249–256, 1992.

A. G. Koru, D. Zhang, K. E. Emam, and H. Liu. An investigation into the functional form of the size-defect relationship for software modules. IEEE Transactions on Software Engineering, 35(2):293–304, 2009.

S. Le Cessie and J. C. Van Houwelingen. Ridge estimators in logistic regression. Applied Statistics, 41(1):191–201, 1992.

K. Lee. Combining multiple feature selection methods. In Mid-Atlantic Student Workshop on Programming Languages and Systems (MASPLAS'02), pages 12.1–12.9, 2002.

S. Lessmann, B. Baesens, C. Mues, and S. Pietsch. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. IEEE Transactions on Software Engineering, 34(4):485–496, 2008.

H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. IEEE Transactions on Knowledge and Data Engineering, 17(4):491–502, 2005.