ISSN (Print): 2476-8316
ISSN (Online): 2635-3490
Dutse Journal of Pure and Applied Sciences (DUJOPAS), *Vol. 5 No. 2b December 2019*

Proposed Defect Modelling for Mitigating Correlated Software Metrics

[1]Shamsuddeen Muhammad Abubakar*,
[2]Zahraddeen Sufyanu

[1,2]Department of Computer Science,
Faculty of Computing
Federal University Dutse,
Jigawa State.
Email: Salsabil012@gmail.com

Abstract
Defect modelling refers to a machine learning model trained using historical project data to estimate the risk of having future defects in modules. The key usage of defect models is in software defect prediction. However, understanding software defect prediction process towards various phases of software practices is a major problem in this field. There are various studies on pitfalls and challenges on defect modelling practice. But, only few researches investigated the modelling process, without paying much attention to choosing different algorithms, constructing and validating defect models. In this paper, various algorithms used in defect prediction models are presented. In addition, software defect collection, defect model construction and model validation are discussed. Meanwhile, literatures related to software defect are highlighted. When defect models were compared, significance results have been reported over previous feature selection techniques. The research suggested a defect model (known as: Embedded Feature Selection) for Correlated Software Metrics Analysis.

Keywords: Defect Models; Feature selection; Software metrics; Correlated Metrics

INTRODUCTION
Defects in code cost industries billions of dollars to find and be corrected. Defect models identify defect prone software modules using a variety of software metrics. They serve two main purposes (Hall *et al.,* 2012; Radjenovic *et al.,* 2013; Shihab, 2012). First, defect models can be used to predict modules that are likely to be defect prone (D'Ambros *et al.,* 2010; Kim *et al.,* 2015; Akiyama *et al.,* 1971). Second, defect models can be used to understand the impact that various software metrics have on the defect proneness of a module (Cataldo *et al.,* 2009; McIntosh*et al.,* 2014; Mockus *et al.,* 2000). The insights derived from defect models can help software teams to avoid the difficulties that led to defective software modules in the past.

Defect models are trained using datasets that connect issue reports recorded in an Issue Tracking System (ITS), with the software modules that are impacted by the associated code changes. The code changes are in turn recorded in a Version Control System (VCS). Thus, the quality of the data recorded in the ITS and VCS impacts the quality of the data used to train defect models (Bird *et al.,*2009; Herzig *et al.,* 2013; Bachmann *et al.,* 2010; Antoniol *et al.,* 2008).

**Author for Correspondence*

The interpretation of defect models heavily relies on the quality of software metrics that are used. However, software metrics often have strong correlation among themselves (Jiarpakdee *et al.,* 2018). The correlated metrics have negative impact on the interpretation of defect models, leading to misleading quality improvement and maintenance plans. To overcome such problems, feature selection techniques are often applied to remove correlated metrics (Jiarpakdee *et al.,* 2018).

Feature selection has been widely used in software engineering to remove irrelevant metrics. These are metrics that do not share a strong relationship with the outcome (Shepperd *et al.,* 2013). Correlated metrics are metrics that shares strong correlation with one or more metrics (Gil *et al.,* 2017). As stated by Lal *et al.* (2006), feature subset selection can be divided into three models: filters, wrappers and embedded.

- Filter Based Method: Is a fast feature selection method, since it does not incorporate learning and rely on the intrinsic characteristics of the training data to select and discard features (Guyon *et al.*, 2002).
- Wrapper Based Method: This involves a learning algorithm (a classifier, or a clustering algorithm) which evaluates each subset of features quality. By including the learning algorithm, accuracy is improved.
- Embedded Based Method: This method embeds feature selection in the training process of the classifier and are usually specific to given learning machines. They are usually faster than both filter and wrapper-based feature selection approaches but are also more likely to overfit (Lal *et al.,* 2006).

Raukas (2017) defined Software metrics as measures of a specific software property. In software defect prediction a set of software metrics are used to extract information about different properties of a software instance such as a file, class and module.

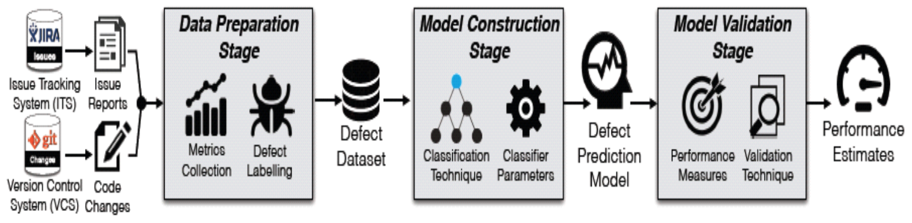
The present study reviews existing literatures across various Software Quality Assurance (SQA),and presents algorithms used in software defect prediction models. The detailed procedures of software defect prediction construction and model validation are analysed. Besides, the study investigates a feature selection using Embedded Based technique to correlate and analyse defect in software metrics analysis. These methods are thus embedded in the algorithm either as its normal or extended functionality. In addition, four criteria in choosing the data sets are followed strictly. Thesedistinguish the present study with many literatures in the field.

PROCESSES INVOLVED IN DEFECT MODEL

There are five main procedures involved in defect model construction. A brief explanation of each process starting from Data Preparation up to Validation is described.

Data Preparation

Module metrics and classes are typically mined from historical repositories, such as Issue Tracking Systems (ITSs) and Version Control Systems (VCSs) (Jeapkadee *et al.,* 2018). Figure 1shows the layout of the defect prediction modelling and its related experimental components.



(Source: Tantithamthavorn, 2017)
Figure 1: An Overview of defect prediction modelling.

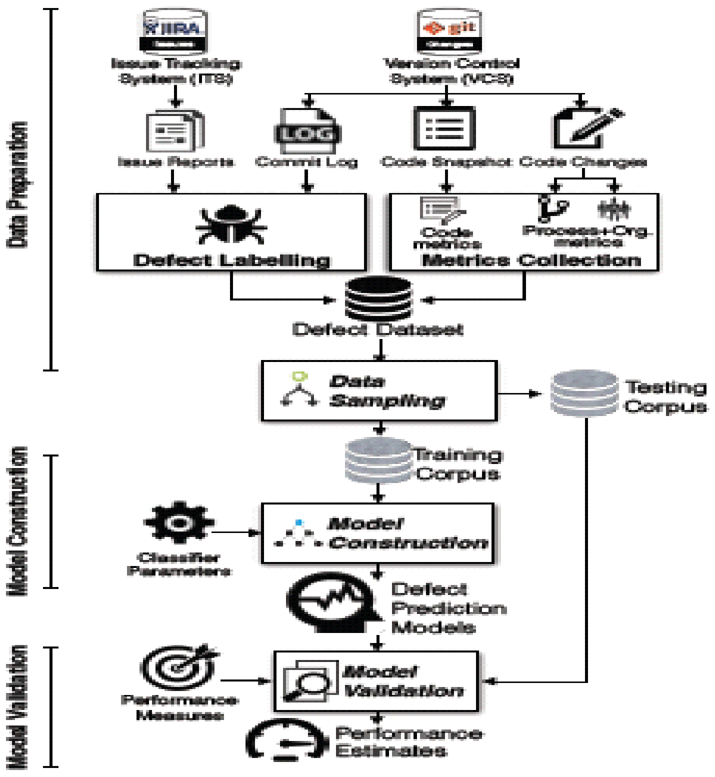
Metric Collection
To understand the characteristics of defect-proneness, previous works proposed a variety of metrics that are related to software quality. They are code metrics, process metrics, and organization metrics (Hoque *et al.,* 2018). Description of the metrics is provided here.

i. **Code Metrics:** This type of software metrics describes the relationship between code properties and software quality. Akiyama *et al.* (1971), is among the first researchers to show that the size of modules e.g., lines of code shares a strong relationship with defect prone in software.
ii. **Process metrics:** This type of metrics describes the relationship between change activities during software development process and software quality. Previous work showed that historical software development process can be used to describe the characteristics of defect proneness (Nagappan *et al.,* 2008).
iii. **Organization metrics:** Organization is a type of metrics which describes the relationship between organization structure and software quality. For example, Graves *et al.* (2000) discovered that modules that are changed by a large number of developers are likely to be more defective. Apart from the organization structure, prior studies have shown that code ownership shares a relationship with software quality.

Defect Labelling
Since modern issue tracking systems such as JIRA, often provides traceable links between issue reports and commit logs, the modules that are changed to address an issue report are identified as either been clean or defective (Tantithamthavorn,2017).

Model Construction
A defect prediction model is a classifier that is trained to identify defect prone software modules. Defect prediction models are trained using the relationship between modules of software metrics extracted from software repositories, and their defectiveness using a statistical regression or a machine learning classifier (Jeapkadee *et al.,* 2018).

Model Validation
Once a defect prediction model is constructed, it is however not known how accurate the defect prediction model will perform when applied to new software modules (i.e., unseen dataset). To address such a problem, model validation techniques (e.g., k-fold cross-validation) are commonly used to estimate the model performance (Okutan *et al.*, 2018). Figure 2 presents the defect prediction modelling process in three stages; data preparation, model construction and model validation.



(Source: Tantithamthavorn, 2017)
Figure 2: Defect prediction modelling process

ALGORITHMS FOR SOFTWARE DEFECT PREDICTION MODELS

Some of the algorithms used within the software defect prediction models is discussed here.

Logistic Regression

Logistic Regression is a machine learning or statistical method used in dataset classification, where the outcome of one or more independent variables is determined. The classification result is the value of one of two possible outcomes (Guyon *et al.,* 2006).

Naïve Bayes

Naïve Bayes is a machine learning algorithm used for classification problems based on Bayes' rule which finds the conditional probability of an instance being labelled with a specific value from the set of labels. The label with the highest probability is considered as the final classification model (Raukas, 2017).

Random Forest

Random Forest is a machine learning algorithm used for classification problem. It consists of the collection of tree predictors and each one is used to classify an unknown instance. The final classification for the unknown instance was determined using the majority result of the trees' predictions (Guyon *et al.,* 2006).

K-Nearest Neighbour

K-Nearest Neighbor is a statistical or machine learning algorithm used on Non-parametric decision procedure which classifies an unknown instance in the category of its nearest neighbour. It uses exhausted search method to find the best candidate for classification (Raukas, 2017).

Support Vector Machine
Support Vector Machine (SVM)is an embedded machine learning method used for classification. When a given set of labelled data which consist of two possible label classes is to be considered, the algorithm constructs a model by mapping the data as points in a space, such that the two separate classes of labelled data are divided by a clear gap which is as wide as possible. The model is then used to map the unknown data into the initial space and that will enable the algorithm to predict the label class of the unknown data based on which side of the gap they are mapped (Guyon *et al.,* 2006).

Artificial Neural Network (ANN)
Artificial Neural Network is statistical or machine learning algorithm used for model classification. The ANN model consists of units of layers called neurons. The layers are three namely, the input layer, hidden layer and output layer. More than one hidden layer can be foundin between the input and output layers. By training the ANN model with a set of data and known labels, the model can learn to predict the values of unknown data (Gotra *et al.,* 2017).

K-Means Clustering
K-Means Clustering is a machine learning or statistical algorithm used to partition a set of data into a specified number of clusters. Each instance from the dataset belongs to the cluster with the nearest mean value (Hoque *et al.,* 2018).

Particle Swarm Optimization
Particle Swarm Optimization is a machine learning algorithm that is used for optimizing parameter values. Particles move around in search of space trying to improve a given measure of quality. Each movement of the particle is influenced by its best known position, it is also been guided towards the best known positions in search of space by other particles. The swarm particle is expected to move towards the best solution (Bowes *et al.,* 2017).

Inter Quartile Range function
Inter Quartile Range function is a statistical algorithm used to detect where bulk of the values lie in a dataset. If the range of the dataset is from the minimum to the maximum value, this method can be used to detect the values ranging from 25% to 75% (Okutan *et al.,* 2018).

Genetic Algorithm
Genetic Algorithm is a machine learning algorithm used for optimization of parameter values. For example, an initial population of candidates to a given solution will randomly be selected from the search space. The population will then generate a better solution by mutating and altering the properties of the candidates (Hoque *et al.,* 2018).

Ensemble Learning
Ensemble Learning is a machine learning algorithm used for statistical analysis to improve the prediction accuracy of machine learning classifiers. Multiple classifiers are trained to solve the same problem and then be combined to form stronger generalization ability (Bowes *et al.,* 2017).

Transfer Learning
Transfer learning is a machine learning algorithm used for statistical analysis which is aimed to transfer the knowledge learned on one dataset, and use that knowledge to solve other problems in a different dataset (Hoque *et al.,* 2018).

Boosting
Boosting is machine learning algorithm used to improve the prediction accuracy of machine learning classifiers by combining set of weak classifiers to create stronger classifiers (Okutan *et al.*, 2018).

Recursive Feature Elimination
Recursive Feature Elimination (**RFE**) is a machine learning algorithm which searches the best subset of metrics by recursively eliminating the least important metrics. Initially, RFE constructs a model using all metrics and ranks the metrics according to their importance score. During the iterations, RFE excludes the least important metric and reconstructs a model. Finally, RFE provides the subset of metrics that yields the best performance according to an evaluation criterion (Jeapkadee *et al.*, 2018).

COMPARISON OF RELATED WORKS
Table 1 presents the summary of the related literatures, the discoveries made and their weaknesses.

Table 1 Summary of Related Literatures

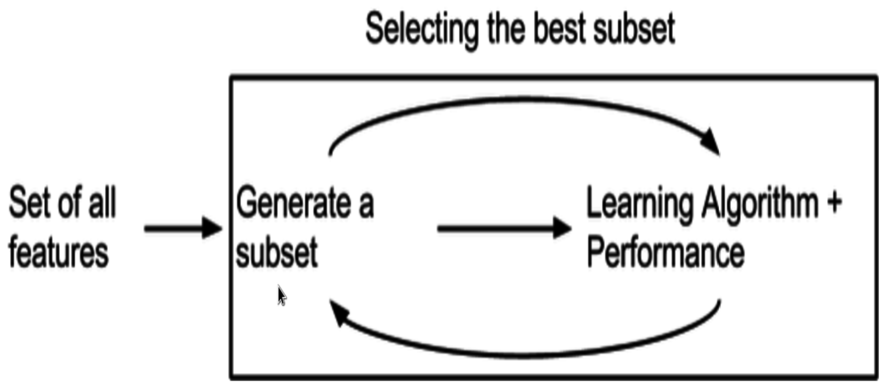
S/N	**Tittle**	**Author**	**Discovery**	**Weakness**
1	A Large-Scale Study of the Impact of Feature Selection Techniques on Defect Classification Models	Ghotra*et al.*,2017.	Looked into Classification Techniques of the defect Models.	Never looked into consistency and correlation of subset of metrics using Embedded feature selection Method.
2	Heterogeneous Defect Prediction	Nam *et al.*, 2017.	How to build Defect prediction in general	Never looked into interpretation of defect models.
3	AutoSpearman: Automatically Mitigating Correlated Software Metrics for Interpreting Defect Models	Jiarpakdee *et al.*, 2018	Looked into two Feature Selection Techniques (Filter and Wrapper based).	Never looked into Embedded based Feature Selection Techniques.
4	Empirical study of feature selection methods over classification algorithms	Bhalaj *et al.*,2018	Investigated the performance of feature selection methods when exposed to different datasets and classification algorithms.	Never looked into mitigation of correlated software metrics.
5	Empirical analysis of change metrics for software fault prediction	Choudhar y *et al.*,2018.	Investigated change metrics in conjunction with code metrics for fault prediction models.	Not investigating the key factors for improving the model performance.
6	The Impact of Correlated Metrics on the Interpretation of Defect Models	Jiarpakdee *et al.*,2018.	Investigated the impact of correlated metrics on the interpretation of defect models and the improvement of the interpretation of defect models when removing correlated metrics.	Tested only Filter and wrapper, no Embedded features election techniques.

7	The Impact of Class Rebalancing Techniques on The Performance and Interpretation of Defect Prediction Models.	Tantitham thavon *et al.,*2018.	Investigated the impact of class rebalancing technique and the interpretation of defect prediction models.	Not improving the efficiency of defect model.
8	The Impact of Correlated Metrics on the Interpretation of Defect Models	Jiarpakdee *et al.,* 2019	Looked into the analysis of Anova datasets and how to select clean datasets for defect prediction.	Nothing was done on embedded and improving the consistency of subset selection.

Embedded methods combine the qualities of filter and wrapper methods. It is implemented by algorithms that have their own built-in feature selection methods (Sandri & Zuccolotto, 2006).In case we have large training set then embedded models can eventually replace filter models (Guyon et al., 2006).

In contrast to filter and wrapper, the learning part and the feature selection part are all combined together in an embedded method (Quinlan, 1992). Decision tree learning can also be considered as an embedded method. The construction of the tree and selection of the features are interleaved. The selection of the feature in each iteration is usually done by a simple filter (Andrew, 2004).

Having found its importance over the two methods further research will deploy the method in detailed. Figure 3 demonstrates how Embedded Feature Selection Technique generates a subset of a metrics from the set of all metrics and combines the performance with the algorithms all in an embedded way.



(Source Guyon *et al.* 2003)

Figure 3: Embedded Feature Selection Technique.

PROPOSED APPROACH

After successfully conducting the reviews on all the related literatures, the research proposes Embedded Feature Selection Techniques using Support Vector Machine(SVM) for Recursive Feature Elimination SVM-RFE for both Random Forest and Logistic Regression (SVM-RFE-RF and SVM-RFE-LR).The algorithms are simple and predominantly used in many machine learning approaches, these prompted us in choosing the two embedded methods.

Previous works used filters such as Correlation based Feature Selection, Information Gain (IG), Chi-Squared base and Consistency based. Others were reported using Wrapper based such Feed-Backward, Feed-Forward and Feed-Both. The current research employs these feature extractions with SVM-RFE-RF and SVM-RFE-LR.

SVM-RFE is an SVM-based feature selection algorithm created by Guyon *et al.* (2003). Initially the algorithm was used for selecting key and important feature sets, apart from reducing classification computational time, it can also improve the classification accuracy rate. The current research will use it in selecting candidate subset for making the consistency and correlation analysis.

As a classification basis, SVM-RFE is an iteration process of the backward removal of features. Its steps for feature set selection are shown as follows:

1. Use the current dataset to train the classifier.
2. Compute the ranking weights for all features.
3. Delete the feature with the smallest weight.

The computational procedures will be presented in steps shown:

(1) Input

- Training sample: $X_0 = [x_1, x_2, \ldots, x_m]^T$.
- Category: $y = [y_1, y_2, \ldots, y_m]^T$.
- The current feature set: $s = [1, 2, \ldots, n]$.
- Feature sorted list: $r = []$.

(2) Feature Sorting

- Repeat the following process until $s = []$.
- To obtain the new training sample matrix according to the remaining features: $X = X_0(:, s)$.
- Training classifier: $a = \text{SVM-train}(X, y)$.
- Calculation of weight: $w = \sum_k a_k y_k x_k$.
- Calculation of sorting standards: $c_i = (w_i)^2$.
- Finding the features of the minimum weight: $f = \arg\min(c)$.
- Updating feature sorted list: $r = [s(f), r]$.
- Removing the features with minimum weight: $s = s(1 : -1, f + 1 : \text{length}(s))$.

(3) Output: Feature Sorted List r. In each loop, the feature with minimum $(w_i)^2$ will be removed. The SVM then retrains the remaining features to obtain the new feature sorting. SVM-RFE repeatedly implements the process until obtaining a feature sorted list. Through training SVM using the feature subsets of the sorted list and evaluating the subsets using the SVM prediction accuracy, we can obtain the optimum feature subsets.

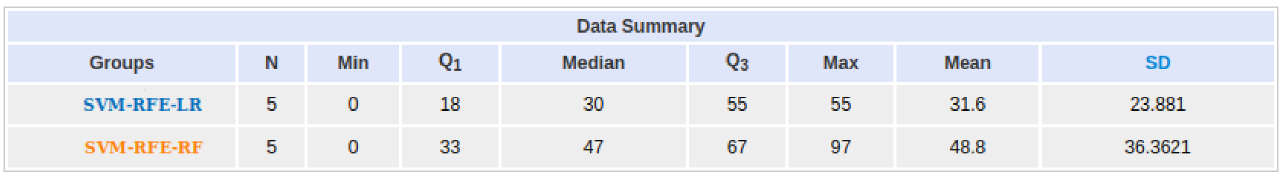
All the experiments will be conducted on the Rstudio platform. Recursive Feature Elimination (RFE) with SVM Feature selection techniques will be used: Random Forest (RF) and Logistic Regression, i.e RFE-RF and RFE-LR.The R packages to be used include Rnalytica, FSelector, sigFeature and caret.

RESULTS AND DISCUSSION

Embedded feature selection techniques produced 15-96% consistent metrics across datasets which is 47% at median. Previous researches on filter and wrapper based feature selection produced up to 41% consistent metrics at median. Figure 3presents the data summary obtained for the two embedded feature selection Techniques.

It is obviously proven that, when SVM with both logistic regression and random forest for recursive feature elimination were deployed, the subset selection of features increases. Therefore, the consistency and correlation of metrics will be better predicted using embedded based feature selection techniques. The present study differs from the work of Tantithamthavon *et al.*, 2018 as it looked into selection of clean datasets for defect prediction, not improving the efficiency of defect model. It also differs from Jiarpakdee *et al.*, 2019 since it considered a barely used approach 'embedded feature selection' and has improved the consistency of subset selection.Subsequent paper following this research will include the detailed results to justify the proposed argument.Also, the research will be limited only to making consistency and correlation analysis. Defect model construction and interpretation will be for further studies.

Table 2: Summary of Embedded based Results



Data Summary								
Groups	N	Min	Q_1	Median	Q_3	Max	Mean	SD
SVM-RFE-LR	5	0	18	30	55	55	31.6	23.881
SVM-RFE-RF	5	0	33	47	67	97	48.8	36.3621

CONCLUSION

Models that can efficiently predict where defects are in code have the capability to save companies large amounts of money. This paper reviewed and summarized different algorithms (used in defect prediction), and methods for construction and validation of defect models. The paper also studied literatures related to software defect with the aim of improving the system. It is found that embedded feature subset selection combined the prediction accuracy for both filter and wrapper methods. To the best of author's knowledge, this method has not been explored by previous researchers in the field. Finding the best candidate for subset selection played a vital role in the construction and interpretation of defect models. Correlated metrics result in negative interpretation of the defect models, this is minimized using consistency of subset selection. The minor results proved that embedded feature selection needs to be employed so as to increase the consistency of the subset selection of the software metrics. Hence, it will improve the mitigation of the correlated software metrics.

ACKNOWLEDGEMENTS

This study would not have been possible without the data shared in the Tera-PROMISE repository used in training and generating results. Prof. Ahmed Baita Garko and Dr. A.M. Miyim have greatly contributed towards making this proposal realistic.

REFERENCES

Ahmet Okutan. (2018). Use of Source Code Similarity Metrics in Software Defect Prediction *arXiv: 1808.10033v1 [cs.SE].*

Akiyama E. (1971). An Example of Software System Debugging. *In Proceedings of the International Federation of Information Processing Societies Congress (IFIP'71), pp. 353–359.*

Andrew Y. Ng. (2004). Feature selection, l1 vs. l2 regularization, and rotational invariance. *In ICML.*

Antoniol G., K. Ayari, M. D. Penta, and F. Khomh. (2008). is it a Bug or an Enhancement? A Text-based Approach to Classify Change Requests. *In Proceedings of the IBM Centre for Advanced Studies Conference (CASCON'08), pp. 1–15.*

Bhalaji,N., K.B. Sundhara Kumar and Chithra Selvaraj. (2018). Empirical study of feature Selection methods over classification algorithms. *In proceedings of international conference of software quality assurance, Australia.*

BirdA., A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov, and P. Devanbu. (2009). Fair and Balanced? Bias in Bug-Fix Datasets. *In Proceedings of the joint meeting of the European Software Engineering Conference and the symposium on the Foundations of Software Engineering (ESEC/FSE'09), pp. 121–130.*

Bowes David, Tracy Hall, and Jean Petri. (2017). Software defect prediction: do different Classifiersfind the same defects? *This article is published with open access at Springerlink.com.*

Cataldo M., A. Mockus, J. Roberts, and J. Herbsleb. (2009). Software Dependencies, Work Dependencies, and Their Impact on Failures. *Transactions on Software Engineering, vol. 35, no. 6, pp. 864–878.*

Choudhary Garvit Rajesh, Sandep Kumar, Kuldeep Kumar, and Alok Mishra. (2018). Empirical Analysis of Change Metrics for Software Fault Prediction. *Computer and Electrical Engineering, Elsevier.com/locate/compeleceng vol. 67, pp. 15-24.*

D'Ambros M., M. Lanza, and R. Robbes. (2010). An Extensive Comparison of Bug Prediction Approaches, *In Proceedings of the Working Conference on Mining Software Repositories (MSR'10), pp. 31–41.*

Ghotra Baljinder, Shane McIntosh, Ahmed E. Hassan. (2017).A Large-Scale Study of the Impact of Feature Selection Techniques on Defect Classification Models.*2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR).*

Graves,T. L. A. F. Karr, J. S. Marron, and H. Siy. (2000). Predicting fault incidence using Software change history. *IEEE Transactions on Software Engineering (TSE), 26(7):653–661.*

Guyon Isabelle and André Elisseeff. (2003). an introduction to variable and feature Selection. *J. Mach. Learn. Res., 3:1157–1182, March 2003.*

Guyon Isabelle, Steve Gunn, Masoud Nikravesh, and Lotfi A. Zadeh. 2006. Feature Extraction: Foundations and Applications (Studies in Fuzziness and Soft Computing). *Springer-Verlag New York, Inc., Secaucus, NJ, USA.*

Hall T., S. Beecham, D. Bowes, D. Gray, and S. Counsell. (2012).A Systematic Literature Review on Fault Prediction Performance in Software Engineering. *Transactions on Software Engineering, vol. 38, no. 6, pp. 1276–1304.*

Herzig K., S. Just, and A. Zeller. (2013). it's Not a Bug, It's a Feature: How Misclassication Impacts Bug Prediction. *In Proceedings of the International Conference on Software Engineering (ICSE'13), pp. 392–401.*

Hoque Nazrul, Mihir Singh, Dhruba K. Bhattacharyya. (2018).EFS-MI: an Ensemble Feature

Selection Method for Classification. *Complex & Intelligent Systems ISSN: 2199-4536.*

Jia Lina. 2018. A Hybrid Feature Selection Method for Software Defect Prediction. *IOP*

Conference Series: Materials Science and Engineering 394.

Jiarpakdee Jirayus, Chakkrit Tantithamthavorn, Christoph Treude. (2018). Autospearman: Automatically Mitigating Correlated Software Metrics for Interpreting Defect Models. *IEEE International Conference on Software Maintenance and Evolution (ICSME).*

Jiarpakdee Jirayus, Chakkrit Tantithamthavorn,and Ahmed E. Hassan. (2018). The Impact of Correlated Metrics on Defect Models. *IEEE Transactions on Software Engineering.*

Jiarpakdee Jirayus, Chakkrit Tantithamthavorn, and Ahmed E. Hassan. (2019). the Impact of Correlated Metrics on the interpretation of Defect Models. *IEEE Transactions on Software Engineering.*

MeneelyA., L. Williams, W. Snipes, and J. Osborne. (2008). Predicting failures with Developer networks and social network analysis. *In Proceedings of the International Sym2posium on the Foundations of Software Engineering (FSE), page 13, New York, New York, USA,*

Nam Jaechang, Weifu, Singhum Kim, Tim Menzies and Lin Tan. (2017). Heterogeneous Defect Prediction. *IEEE Transaction on Software Engineering.*

Nagappan N., B. Murphy, and V. Basili. (2008). the influence of organizational structure on Software quality: an empirical case study. *In Proceedings of the International Conference on Software Engineering (ICSE), pages 521–530. ACM.*

Osman Haidar, Mohammad Ghafari, Oscar Nierstrasz. (2018). the Impact of Feature Selection on Predicting the Number of Bugs. *arXiv: 1807.04486v1 [cs.SE].*

Radjenovic D., M. Hericko, R. Torkar, and A. Zivkovic. (2013). Software Fault Prediction Metrics: A Systematic Literature Review. *Information and Software Technology, vol. 55, no. 8, pp. 1397–1418.*

Raukas Hans. (2017). Some Approaches for Software Defect Prediction. PhD thesis, UNIVERSITY OF TARTU Institute of Computer Science Computer Science Curriculum.

Quinlan,J. Ross. (1999). C4.5: Programs for Machine Learning. (Morgan Kaufmann Series in Machine Learning). Morgan Kaufmann, 1 edition, Oct 1992.

Sandri, M. and P. Zuccolotto (2006). Variable Selection Using Random Forests. *Springer, pp. 263–270.*

Shihab. (2012). an Exploration of Challenges Limiting Pragmatic Software Defect Prediction. *Ph.D. dissertation, Queen's University.*

TantithamthavornChakkrit, Ahmed E. Hassan. (2018). an Experience Report on Defect Modelling in Practice: Pitfalls and Challenge. 2018. *IEEE/ACM 40th International Conference on Software Engineering in Practice Track (ICSE-SEIP).*