FUDMA Journal of Sciences (FJS)
ISSN online: 2616-1370
ISSN print: 2645 - 2944
Vol. 4 No. 1, March, 2020, pp 62 – 68

A SURVEY OF FEATURE SELECTION METHODS FOR SOFTWARE DEFECT PREDICTION MODELS

[1]Shamsuddeen M. Abubakar , [2]Zahraddeen Sufyanu, [3]Abubakar M. Miyim

[1, 2, 3] Department of Computer Science, Faculty of Computing Federal University Dutse, Jigawa State.

Corresponding authors email: Salsabil012@gmail.com, sufyanzzzzz@gmail.com, abubakar.miyim@fud.edu.ng

ABSTRACT
Feature selection is a technique used to select an optimal feature subset from the original input features according to a specific criterion. The criterion is often formulated as an objective function that finds which features are most appropriate for some tasks at hand. The reason why it is interesting to find a subset of features is because that it always easier to solve a problem in a lower dimension. This helps in understanding the nonlinear mapping between input and output variables. This paper reviewed the basic Feature Selection Techniques for Software Defect Prediction Model and their domain applications. The Subsets selection are categorized into three distinct models and are discussed in a concise form to provide young researchers with the general methods of Subset Selection. Support Vector Machine with Recursive Feature Elimination for both Logistic Regression and Random Forest was introduced to evaluate the performance between filter, wrapper, and embedded feature selection techniques. Hence, the research proposes an Embedded Feature Selection techniques for consistency of a subset of software metrics analysis.

Keywords: Defect Models; Defect Prediction; Feature selection; Software Metrics

INTRODUCTION

Defect prediction is the process of determining which parts of a software system may contain defects (Kitchenham, 2007).

Defect prediction models are trained using historical data to identify defect-prone software modules (Tantitthamavorn *et al.,* 2017). From a Software Quality Assurance (SQA) perspective, defect prediction models serve two main purposes;

a. It can be used to predict modules that are likely to be defect-prone in the future. Therefore, SQA teams can use defect prediction models in a prediction setting to effectively allocate their limited resources to the modules that are most likely to be defective.

b. It can be used to understand the impact of various software metrics on the defect-proneness of a module. The insights that software teams derive from defect prediction models can help them avoid past pitfalls that are associated with the defective modules of the past (Tantitthamavorn *et al.,* 2018).

Defect models are statistical or machine learning models that are used to investigate the impact of software metrics on defect-proneness and to identify defect prone in the software modules (Jiarpakdee *et al.,* 2018).

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 e.g. a file, class, and module. However, software metrics often have a strong correlation among themselves (Jiarpakdee *et al.* 2018). The construction of defect models heavily relies on the quality of software metrics that are used.

Previous works found that many metrics in defect datasets are correlated e.g. the branch count metric is linearly proportional to the decision count metric in some NASA datasets (Jiarpakdee *et al.,* 2018).

Thus, correlated metrics will make the construction of defect models gloomy. As such, the most important characteristics of defective modules that are derived from defect models may be incorrect, 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).

The present study reviews existing literatures in the field of software defect prediction, research gap was discovered and a new algorithm was deployed to increase the efficacy in the subset of metrics selection. Preliminary results will be presented.

FEATURE SELECTION TECHNIQUES

Feature selection is a data preprocessing technique for selecting a subset of the best software metrics before the construction of a defect model. Feature selection has been widely used in software engineering to remove irrelevant metrics (i.e., metrics that do not share a strong relationship with the outcome) and correlated metrics (Jiarpardee *et al.,* 2018).

Feature selection is the mechanism of selecting a subset of features from the given set of features to reduce the redundant and irrelevant features without much loss of the facts and details (Hoque *et al.,* 2018).

According to Lal *et al*. (2006) the purpose of feature selection is three-fold.

i. Lower dimensionality enhances the prediction accuracy of a classifier. Once an optimal subset has been determined, a simple learning algorithm can give a very good performance.

ii. Most of the learning algorithms become computationally intractable when the number of features is large, both in the training and in the prediction step. A step of feature selection preceding the training algorithm can alleviate the computational burden.

iii. Dimensionality reduction provides better penetration into the process that generated the data. This purpose is substantial since in many cases the ability to point out the most informative features is important.

The selection of features can be achieved in two ways, feature ranking and features subset selection (Raukas, 2017). Detailed information will be provided in the subsequent subheadings.

a) Ranking

In feature ranking approach, features are ranked according to some criterion to select the top "*n*" ranked features based on their relevance and this number "*n*" is either specified by the user or determined automatically (Okutan *et al.*, 2018). The main drawback of this method is that it assumes the features to be independent of each other. This causes two problems: Features that are discarded for not being relevant may turn relevant when compared with other features. Besides, features that are regarded individually relevant may cause unnecessary redundancies.

Feature ranking uses scoring functions (Euclidean distance), correlation (Pear-son correlation coefficient) or information-based criteria for evaluation. Generally, this is used as a preprocessing step as it is usually efficient from the computational point of view. Conversely, the technique inevitably fails in situations where only a combined set of features is assumed to be predictive of the target function. This technique normally fits problems such as micro-array analysis (Jeapkadee *et al.,* 2018).

b) Subset Selection

In contrast to feature ranking, feature subset selection algorithms may automatically find how many features have to select. The rapid advances in several research fields with huge datasets made it essential to select only the most important or descriptive features and the remaining are discarded (Gotra *et al.,* 2017).

According to the work of Bowes *et al.* (2017), feature subset selection approach is believed to have better predictive ability than that of feature ranking according to their predictive power. As stated, a single feature that is completely useless by itself can strikingly improve performance when taken in account with other features. On the other hand, a good feature that is highly correlated with another feature already in the subset would provide no additional benefit since it would be redundant. Feature ranking approaches cannot manage to deal with these scenarios. Figure 1 shows an overview of the feature selection phase.

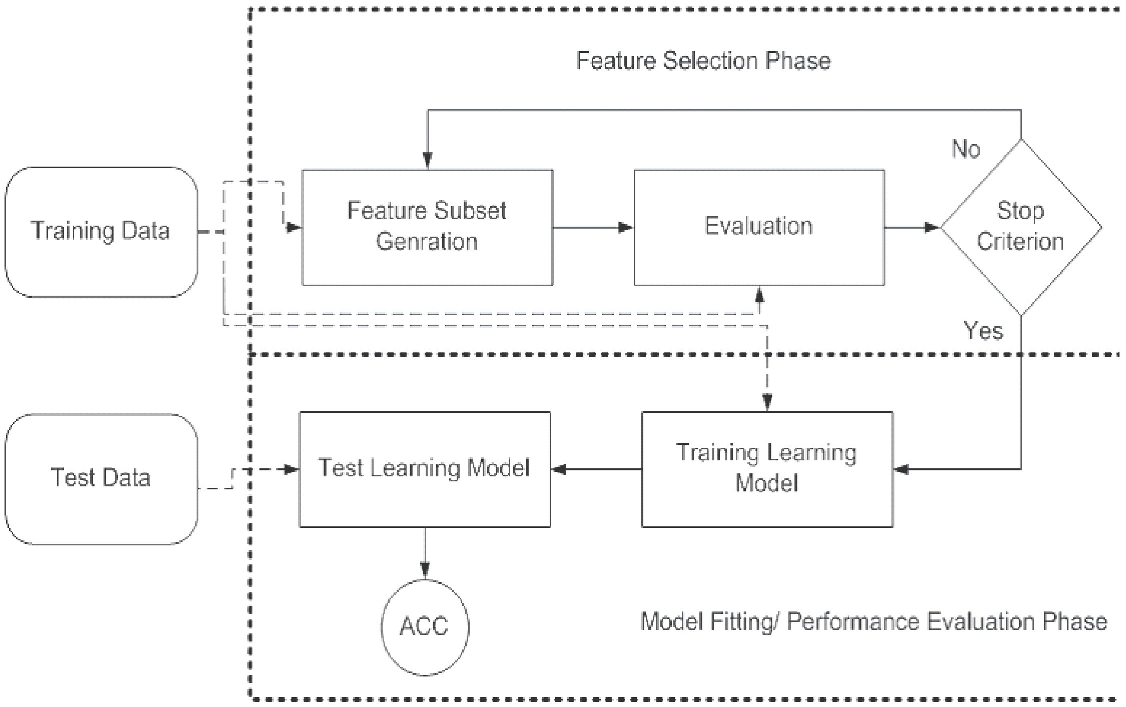


Fig. 1: Feature Selection Phase

(Source: Lal *et al.*, 2006)

Feature subset selection can be divided into three models: filters, wrappers and embedded. All feature selection models have their advantages and drawbacks (Lal *et al.,* 2006).

A. Filter Methods

Filter methods were the earliest approaches for feature selection in machine learning approach. Filters are algorithms the filter out insignificant features that have little chance to be useful in the analysis of data. They are fast due to the fact they do not incorporate learning and rely on the intrinsic characteristics of the training data to select and discard features (Tantithamthavon *et al.,* 2018).

The filter methods are algorithms with relevance index J(S|D, Y), that approximates a given data D, how relevant a given feature subset S is for task Y. These indices are commonly known as feature selection metrics. Some popular Metrics are correlation-based, distance-based, information-based, and some algorithmic procedures may be used to estimate the relevance index such as decision trees (Tantithamthavon *et al.,* 2018). Figure 2 demonstrates the filter-based feature selection techniques and how it performs its operation.

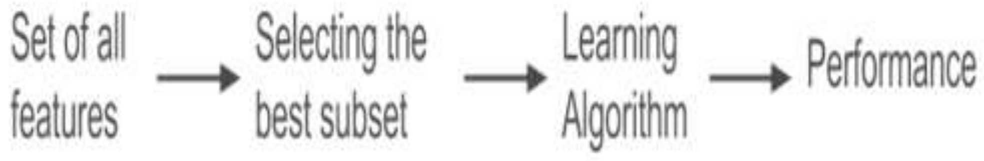


Fig. 2: Filter based Feature Selection Techniques
(Source: Guyon *et al.,* 2012)

Filter methods tend to select subsets with a high number of features and therefore a proper threshold is required to choose a subset. Concerning feature selection metrics, in contrast to information-based and decision tree based metrics, correlation coefficient approach is perhaps the simplest and preferable, because it avoids problems with probability density estimation (Gotra *et al.,* 2017).

B. Wrapper Methods

Wrapper methods select a feature subset using a learning algorithm as part of the evaluation function. The learning algorithm is used as a kind of "black box" function to guide the search. The evaluation function for each candidate feature subset returns an estimate of the quality of the model that is induced by the learning algorithm, which therefore causes a better estimate of accuracy (Choudhary *et al.,* 2018).

Wrapper methods tend to be prohibitively slow and computationally expensive, since, for each candidate feature subset evaluated during the search, the target learning algorithm is usually applied several times. In wrapper methods, a search strategy iteratively adds or removes features from the data to search the best possible features subset that maximizes accuracy. A search approach decides the order in which the variable subsets are evaluated such as best-first, exhaustive search, simulated annealing, genetic algorithms, branch and bound (Choudhary *et al.,* 2018). Figure 3 presents Wrapper-based Feature Selection Techniques and how it performs it is operations.

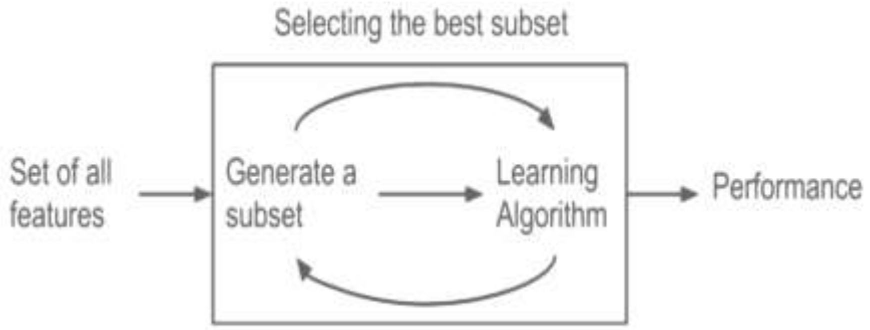


Fig. 3: Wrapper based Feature Selection Technique
(Source: Guyon *et al.,* 2012)

Embedded Methods

In contrast to filter and wrapper, the learning part and the feature selection part are carried out together in an embedded method. Decision tree learning can also be considered to be an embedded method, as the construction of the tree and the selection of the features are interleaved. The selection of the feature in each iteration is usually done by a simple filter or ranker. A well-known example of an embedded method is the L1-SVM (Bowes *et al.,* 2017).

Embedded methods combine the qualities of filter and wrapper methods. It's implemented by algorithms that have their built-in feature selection methods. Embedded methods perform feature selection during the modeling algorithm's execution. These methods are thus embedded in the algorithm either as its normal or extended functionality. Common embedded methods include various types of decision tree algorithms: CART, C4.5, random forest, but also other algorithms (e.g. multinomial logistic regression and its variants) (Bhalaj *et al.,* 2018).

Some embedded methods perform feature weighting based on regularization models with objective functions that minimize fitting errors and in the meantime force the feature coefficients to be small or to be exactly zero. These methods usually work with linear classifiers like Support Vector Machine SVM and induce penalties to features that do not contribute to the model (Bhalaj *et al.,* 2018).

An embedded model embeds feature selection in the training process of the classifier and is usually specific to given learning machines. They are usually faster than wrapper approaches but are also more likely to overfit (Bhalaj *et al.,* 2018). Figure 4 demonstrates how Embedded Feature Selection Technique performs its operation.

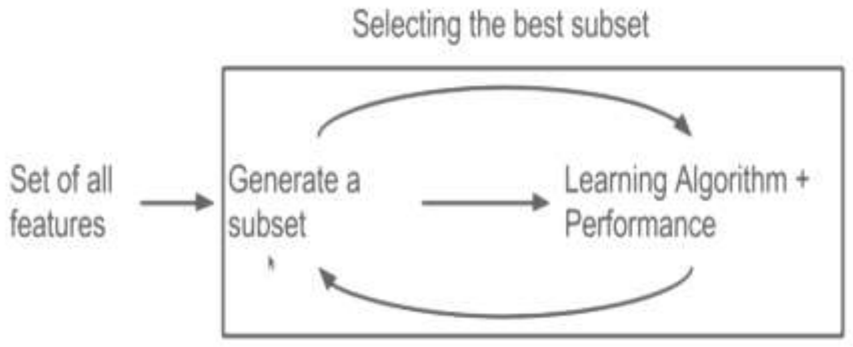


Fig. 4: Embedded Feature Selection Technique
(Source: Guyon *et al.,* 2012)

Feature Selection Domain Application

The choice of feature selection methods differs among various application areas (Jović *et al*. 2015). A brief description of the domain areas of application of feature selection techniques are discussed here.

i. Text Analytics

Text Analytics is the standard way of representing a document by using the bag-of-words model. The idea is to model each document with the counts of words occurring in that document. Feature vectors are typically formed so that each feature (i.e. each element of the feature vector) represents the count of a specific word, an alternative approach is to indicate the presence or absence of a word without specifying the count. The set of words whose occurrences are counted is called a vocabulary. Given a dataset that needs to be represented, one can use all the words from all the documents in the dataset to build the vocabulary and then prune the vocabulary using feature selection (Jović *et al.,* 2015).

ii. Bioinformatics

One of the interesting application of feature selection is in biomarker discovery from genomics data. In genomics data, individual features correspond to genes, so by selecting the most relevant features, one gains important knowledge about the genes that are the most discriminative for a particular problem (Jović *et al.,* 2015).

iii. Industrial applications

Feature selection is important in fault diagnosis for industrial applications, where numerous redundant sensors monitor the performance of a machine (Jović *et al.,* 2015). Liu *et al.* (2014) have shown that the accuracy of detecting a fault (i.e. solving a binary classification problem of machine state as defective or non-defective) can be improved by using feature selection.

iv. Image Processing and Computer Vision

Representing images is not a straightforward task, as the number of possible image features is practically unlimited (Bins *et al.,* 2015). The choice of features typically depends on the target application. Examples of features include histograms of oriented gradients, edge orientation histograms, Haar wavelets, raw pixels, gradient values, edges, and color channels. (Jović *et al.,* 2015).

METHOD AND TOOLS USED

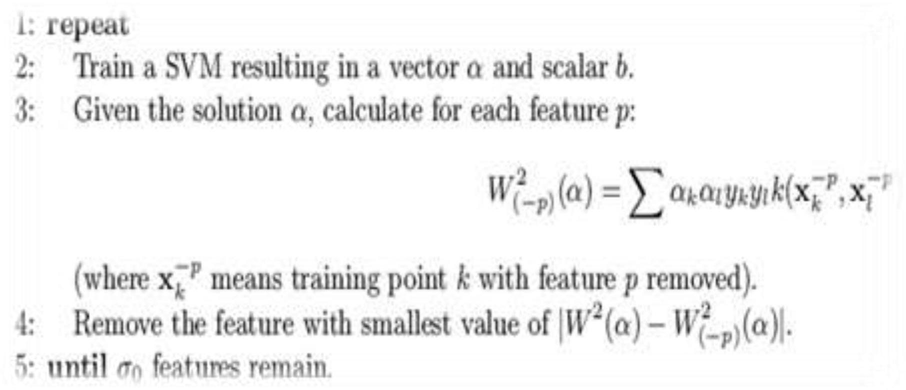
Recursive Feature Elimination (RFE) is a feature selection algorithm, which employs only $\sigma_0 < n$ input dimensions in the final decision rule, it finds the best subset of size σ_0 by a kind of greedy backward selection

It operates by trying to choose the σ_0 features which lead to the largest margin of class separation, using an SVM classifier. This combinatorial problem is solved in a greedy fashion at each iteration of training by removing the input dimension that decreases the margin until only σ_0 input dimensions remain.

The algorithm can be accelerated by removing more than one feature in step 2. RFE has shown good performance on problems of gene selection for microarray data.

The algorithm can be generalized as a nonlinear case. For Support Vector Machine (SVM) the margin P is inversely proportional to the value $W^2(\alpha) := \sum \alpha_k \alpha_l y_k y_l k(\mathbf{x}_k, \mathbf{x}_l)$ $(= \|\mathbf{w}\|^2)$. The algorithm thus tries to remove features which keep this quantity small. This leads to the following iterative procedures. Figure 5 presents the algorithm used in the proposed method.

Procedures:



```
1: repeat
2:    Train a SVM resulting in a vector $\alpha$ and scalar $b$.
3:    Given the solution $\alpha$, calculate for each feature $p$:
          $W^2_{(-p)}(\alpha) = \sum \alpha_k \alpha_l y_k y_l k(\mathbf{x}_k^{-p}, \mathbf{x}_l^{-p})$
      (where $\mathbf{x}_k^{-p}$ means training point $k$ with feature $p$ removed).
4:    Remove the feature with smallest value of $|W^2(\alpha) - W^2_{(-p)}(\alpha)|$.
5: until $\sigma_0$ features remain.
```

Fig.5: Non-linear Recursive Feature Elimination Algorithm
(Source: Guyon *et al.,* 2012)

All the experiments will be performed on the Rstudio platform, PSPP, Past325, and Microsoft Excel. 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, while PSPP will be used for data analysis, Past325 will be used for boxplot and finally Microsoft excel for percentage computations. Table 1 presents the general overview of the implementation of an embedded feature selection technique.

Table 1: The Embedded Feature selection Technique

Method	Technique	R Package	Functions	Abbreviation
Embedded	Recursive Feature Elimination (Random Forest) with SVM	SigFeature	svmrfeFeature Ranking(*x,y*) summary(*RFE.*	SVM-RFE-RF
	Recursive Feature Elimination (Logistic Regression) with SVM	Caret Rnalytica	*model*) rfe step	SVM-RFE-LR

RESULTS AND DISCUSSION

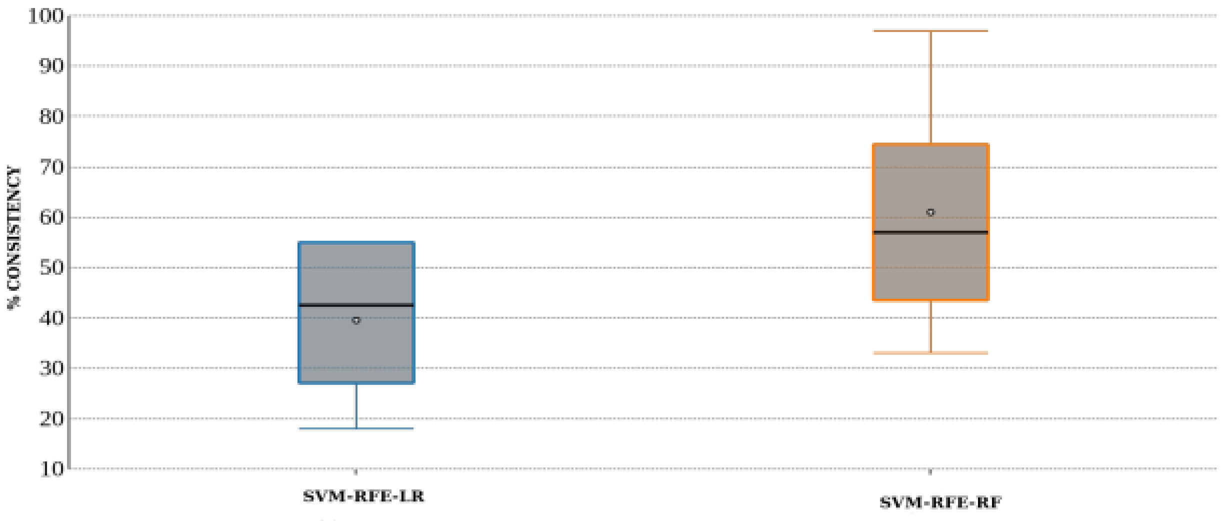
Embedded Feature Selection Technique produced 18-55% consistent metrics across datasets. Were the previous research on Filter and Wrapper Selection techniques produced only 6% consistent metrics at a median.

It is evidently proven that embedded feature subset selection combines the prediction accuracy of both filter and wrapper-based feature selection techniques. Table 2 presents a general summary of the results obtained. The datasets used for training are divided into four categories.
Category 1: Are the datasets which includes Eclipse 2.0, Eclipse 2.1 and Eclipse 3.0.
Category 2: Are the datasets which includes the EclipseJDT and EclipseMylyn.
Category 3: Are the datasets which includes EclipseDebug3.4 and EclipseSWT3.4.
Category 4: Are the last set of the datasets which includes Prof 1, Prof 2 and Xylan 2.6.

Table 2: Summary of Generated Results

Category	Intersection	Union	Percentage
Category1	11	20	55
Category2	2	11	18.18181818
Category3	3	10	30
Category4	11	20	55

Figure 5 presents the boxplot of the results obtained for the two embedded feature selection techniques used in the research.
Fig. 6: % Consistency against Two Embedded Feature Selection Techniques



CONCLUSION

The research investigated various techniques to feature selection. The three most important Feature selection techniques are filter, wrapper, and embedded-based techniques. The domain applications to feature selection was highlighted. Support Vector Machine with Recursive Feature Elimination for both Random Forest and Logistic Regression was employed as the two embedded method used in the research. The preliminary results obtained proved that, embedded feature selection techniques increases the consistency of subset of software metrics selection. The two commonly used techniques discussed by previous researchers were found to have many drawbacks, that is what make it eminent to further the research on embedded-based method which the feature work will look into. Having studied that the embedded feature selection techniques combines the qualities of both filter and wrapper techniques it is clear that a research needs to be conducted on it, and to compare the efficiency of consistency and correlation between the techniques. Three criteria will duly be considered in data collection. Lastly, embedded feature selection technique will be investigated and to identify how it is used for correlated software metrics analysis.

ACKNOWLEDGEMENTS
The authors acknowledged Prof. Ahmed Baita Garko who have greatly contributed towards making the research a realistic.

Lastly to the reviewers for their positive contribution which improved the manuscript.

REFERENCES

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.*

Bins J. and B. A. Draper. (2015). Feature selection from huge feature sets," *in Proc. 8th International Conference on Computer Vision (ICCV-01), Vancouver, British Columbia, Canada, IEEE Computer Society: 159–165.*

Bowes David, Tracy Hall, and Jean Petri. (2017). Software defect prediction: do different Classifiers find the same defects? *This article is published with open access at Springerlink.com.*
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, 67:15-24.*

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).*

Hoque Nazrul, Mihir Singh, Dhruba K. Bhattacharyya. (2018). EFS-MI: an Ensemble Feature Selection Method for Classification. *Complex & Intelligent Systems ISSN: 2199-4536.*

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

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

Jović A., K. Brkić, and N. Bogunović. (2015). A review of feature selection methods with Applications. *763354 MIPRO.*

Lal Thomas, Olivier Chapelle, Jason Weston, and AndrÃl' Elisseeff. (2006). Embedded Methods in Feature Extraction. *Volume 207 of Studies in Fuzziness and Soft Computing, pages 137–165. Springer Berlin Heidelberg.*

Liu C., D. Jiang, and W. Yang. (2014). Global geometric similarity scheme for feature selection in fault diagnosis. *Expert Systems with Applications. 41(8): 3585–3595.*
Kitchenham, B.A (2007). Guidelines for Performing Systematic Literature Review in Software Engineering; Technical Report. *EBSE-2007-001; Keele University and Durham University: Staffordshire, UK.*

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

Raukas Hans. (2017). Some Approaches for Software Defect Prediction. Ph.DDissertation *UNIVERSITY OF TARTU Institute of Computer Science Computer Science Curriculum.*

Tantithamthavorn Chakkrit, Shane Mcintosh, Ahmed E. Hassan, Kenichi M. (2018). The Impact of Automated Parameter Optimization on Defect Prediction Models. *IEEE Transaction on Software Engineering.*