YUMSUK
JOURNAL OF PURE AND APPLIED SCIENCES

COMPARISONS OF DIMENSIONALITY REDUCTION TECHNIQUES FOR SOFTWARE DEFECT PREDICTION

[1]Shamsuddeen M. Abubakar, [2]Abdulmajid B. Umar, [3]Ahmed B. Garko, [4] Zahraddeen Sufyanu, [5]Abdullahi M. Ibrahim, [6]Sagiru Mati and [7]Mohammed K. Dauda
[1,3,4,5]Department of Computer Science, Faculty of Computing Federal University Dutse, Jigawa State.
[2]Department of Computer Science, Faculty of Computing Yusuf Mai-Tama Sule University, Kano.
[6]Operational Research Center in Healthcare, Near East University, North Cyprus
[7]Department of Computer Science, Abubakar Tafawa Balewa (ATBU) Bauchi.
*Corresponding author's e-mail: salsabil012@gmail.com

Abstract

One of the most important roles of a software engineer is the software quality assurance (SQA) activities. The major activities of software engineers are to release defect-free software to end consumers. However, software programs contain large and delicately correlated software metrics that are found in different software modules. These made the deployment of the software defect prediction model more complex. The objective of this research is to examine the consistency and correlation of subsets of metrics that are generated by feature selection techniques on 10 publicly available defect data sets. The research found that the correlation-based feature selection approach of the feature selection technique produced the best prediction accuracy of 85.20% with a 0.87 F1 score measure and a 0.92 AUC. The consistency of a subset of metrics was measured by the percentage of metrics that were consistently selected across different training samples from the same dataset. The best way to choose a feature selection technique is to experiment with different techniques and evaluate their performance on a specific dataset and machine learning model. The research findings have shown that the best method was **Correlation-based feature selection (CFS)**, which provided the highest accuracy, F1 score, and AUC and can be used to improve the performance of software defect prediction models by reducing the computational complexity of software metrics selection. The feature selection technique-based approach for software defect prediction has shown significant enhancement, and we recommend that future studies perform software defect model construction using different datasets to justify the study.

Keywords: Dimensionality Reduction Techniques, Software Defect Prediction, Software Metrics, Correlated Metrics, Computational Complexity

INTRODUCTION

Feature reduction techniques was used as a data pre-processing technique to find a candidate subset of software metrics prior to developing a software defect prediction model (Menzies, 2018). In software quality assurance (SQA), its algorithms have been thoroughly studied to remove redundant software metrics. According to Shepperd *et al.* (2013), software metrics are those that have no impact on the software defect model's overall performance.

Software engineers uses software metrics to estimate software development time and evaluate the overall qualities of the program (Fenton *et al.*, 1999). It was applied to enhance the defect prediction model's software features. To extract the properties of different software assets, such a file, class, or module, software metrics are widely utilized. There are condensed and more complicated software measurements. The Lines of Code (LOC) metric is a commonly used tool in software quality assurance (SQA) to calculate the total number of lines of code in a software instance.

A coding error is referred to as a software flaw. These errors may result from software products that fall short of end users' expectations or from other unmet software requirements. The software product could display an unanticipated behavior as a result of the defect, leading to a complete program failure (Abubakar *et al.*, 2021). The essential ideas that define the software defect are as follows:

a. Defect is a bug that results from a programming error that becomes incorporated into an application. A software product is deemed defective if its output deviates from the anticipated outcome specified in the software specification document.

b. A software product may be deemed defective if it is unable to fulfill the needs or expectations of the end user. This unhappiness could result from a mistake made in the product's development process or in its procedures.

Models for predicting software defects played a crucial role in developing SQA initiatives. Software metrics are used to train software defect prediction models, which then identify the software modules that are prone to defects (Tantitthamavorn *et al.*, 2018).

These two crucial tasks are carried out by the software defect prediction model (Akiyama, 1971) and are explained below.

Software modules that are likely to have issues are identified using software defect prediction models (Akiyama, 1971). Therefore, in order to effectively devote specific resources to the modules that are most likely to be defective, SQA engineers employ models to analyze the settings (D'Ambros *et al.*2010).

To evaluate the impact of different software metrics on software modules that are prone to defects, software defect models are employed (Cataldo *et al.*, 2009). To prevent past mistakes related to software modules that are prone to defects, the SQA team makes use of the intuition obtained from software defect prediction models (McIntosh *et al.*, 2014).

Defect prediction models have become more widely used in practice throughout the past ten years. Defect prediction technique has recently been adopted by Bell Labs, AT&T, Turkish Telecommunication, Microsoft Research, Google, Blackberry, Cisco, IBM, and Sony Mobile. According to Tantitthamavorn *et al.* (2018), these businesses have effectively incorporated defect prediction techniques and gained insightful knowledge.

One important method in the realm of software defect prediction is dimensionality reduction. By lowering the amount of input variables, it aids in the improvement of prediction models, enhancing interpretability, performance, and reducing overfitting. Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and feature selection techniques are common approaches for reducing dimensionality. However, practitioners find it difficult to select the best appropriate technique from among them. This study examines how well five distinct dimension reduction strategies perform in a software defect prediction model.

RELATED WORKS

In software defect prediction (SDP), dimensionality reduction has emerged as a key strategy for improving the precision and effectiveness of predictive models. To reduce problems that can negatively affect the performance of machine learning algorithms, such as overfitting, high computing cost, and the curse of dimensionality, this procedure entails lowering the amount of input variables (Zhang *et al.*,2020).

Techniques for reducing dimensionality aid in streamlining the data structure, making it easier to handle and understand. Repetitive and unnecessary features are frequently present in high-dimensional data, which can make it difficult to see the underlying patterns required for precise fault prediction. Computational complexity of software defect prediction model can be decreased, and its performance can be enhanced by lowering the dimensionality (Zhang *et al.*,2020).

Das (2022) investigated the benefits and drawbacks of both filter-based and wrapper-based feature selection for feature selection and proposed a hybrid algorithm that employs boosting and incorporates some of the features of wrapper methods into the filter method for feature selection. The study did well in investigating the differences

between filter and wrapper, resulting in a hybrid algorithm, and the results showed that wrapper is faster than filter feature selection techniques, despite the fact that the embedded method is faster than both filter and wrapper methods.

Also lai and Roper (2019) conducted a systematic review of the use of feature selection strategies in software quality prediction. The authors discovered that feature selection strategies can increase the performance of software quality prediction models, with filter-based methods being the most widely utilized.
Kakkar and Jain's (2020) research, titled Feature Selection in Software Defect Prediction: A Comparative Study, aimed to provide a framework for defect prediction based on five classifiers. The authors discovered that feature selection can increase the performance of defect prediction models, and the Relief F algorithm was the most effective feature selection method.

Liu *et al.* (2021) paper, "FECAR: A Feature Selection Framework for Software Defect Prediction," introduced a feature clustering and ranking technique (FECAR) and an FF-correlation metric for prediction. The authors discovered that FECAR performed better than other feature selection strategies in terms of prediction accuracy.
Thirumoorthy and Britto's (2022) paper, "A feature selection model for software defect prediction using the binary Rao optimization algorithm," proposed a hybrid feature selection (filter-wrapper) approach based on the multi-criteria decision making (MCDM) method and the Rao optimization algorithm. The authors discovered that their strategy outperformed previous feature selection techniques in terms of predictive accuracy.

Work of Balogun *et al.* (2019) titled: Performance Analysis of Feature Selection Methods in Software Defect Prediction: A Search Method Approach, analyzed the performance of different feature selection methods in software defect prediction. The authors found that Consistency Feature Subset Selection based on Best First Search had the best influence on the prediction models.
Balogun *et al.* (2021) study, Software Defect Prediction Using Wrapper Feature Selection Based on Dynamic Re-Ranking Strategy, offered a wrapper feature selection approach based on a dynamic re-ranking strategy. The authors discovered that their strategy outperformed previous feature selection techniques in terms of predictive accuracy.

Abubakar *et al.* (2020) examined the fundamental feature selection methods for software defect prediction models and their domain applicability. When analyzing the performance of filter, wrapper, and embedding feature selection strategies, Support Vector Machines with Recursive Feature Elimination were used for both Logistic Regression and Random Forest. The researchers propose employing embedded feature selection approaches to correlate a subset of software metrics.

3. MATERIALSANDMETHOD

In order to test the model, we used a feature selection technique-based method for software defect prediction on ten publicly accessible defect datasets. A rudimentary learner was implemented utilizing more sophisticated methods, such as J48, while model trees will be utilized to anticipate problems. The experiment is carried out in Python with the help of Anaconda Navigator.

1. Data extraction

In the research, the datasets were mined from different repositories and open-source domains after critical analysis and applying criteria in selecting them; their descriptions are shown in Table 1.

Table 1: *The studied datasets*

Dataset	Modules Count	Metrics	Defective Ratio %	EPV	AUC_{LR}	AUC_{RF}
Xalan2.6	889	22	45	21	0.79	0.86
Debug 3.4	1,165	18	27	15	0.72	0.83
JDT	1,097	17	24	14	0.81	0.81
Mylyn	1,952	17	17	16	0.78	0.76
Platform 2	6,989	36	18	30	0.82	0.82
Platform 2.1	8,788	36	16	27	0.77	0.72
Platform 3	11,593	36	14	49	0.79	0.85
SWT 3.4	1,485	19	48	38	0.87	0.95
Prop 1	18,471	23	17	137	0.75	0.76
Prop 2	23,014	28	13	122	0.71	0.85

Guyon and Elisseeff (2003) define the three stages of selecting the optimal feature sets as follows:

i. Classification of input datasets
ii. Calculating feature weight and
iii. The minimal weight feature is removed in order to gain feature ranking.

2. The experimental steps are shown as follows:

(a) The Input Datasets

Training sample $\boldsymbol{X_0 = [x_1, x_2 \ldots\ldots\ldots x_m]T} \ldots\ldots \boldsymbol{i}$
Category $\boldsymbol{Y = [y_1, y_2 \ldots\ldots\ldots y_m]T} \ldots\ldots\ldots\ldots \boldsymbol{ii}$
Current feature sets $\boldsymbol{S = [1, 2, 3 \ldots\ldots\ldots n]} \ldots \boldsymbol{iii}$
List of Sorted Features $\boldsymbol{r_0 = [\ ]} \ldots\ldots\ldots\ldots\ldots\ldots \boldsymbol{iv}$

(b) Sorted Features

The iteration process is continuing until all the features are sorted $\boldsymbol{s = [\ ]} \ldots\ldots\ldots\ldots\ldots\ldots\ldots \boldsymbol{1}$

The new training sample is generated by applying the matrix by retaining the rest of the features $\boldsymbol{X = X_0(s)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \boldsymbol{2}$

Training the classifier $\boldsymbol{\alpha = SVM - train(X, Y)} \ldots \boldsymbol{3}$
Weight calculation $\boldsymbol{W = \sum K\alpha kYkXk} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \boldsymbol{4}$
Standard for sorting $\boldsymbol{C_i = (W_i)2} \ldots\ldots\ldots \boldsymbol{5}$
Minimum feature weight $\boldsymbol{f = arg.min(c)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \boldsymbol{6}$

The updated feature sorted list $\boldsymbol{r = [S(f)r]} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \boldsymbol{7}$

Removing the feature with minimum weight $\boldsymbol{S = s(1: -1, f1: length(s))} \ldots\ldots\ldots\ldots \boldsymbol{8}$

(c) Sorted Feature List Output r.

From each loop, the feature with minimum weight $(Wi)2$ will be removed.

Dimensionality Reduction Techniques

a. Principal Component Analysis (PCA)

An approach that is frequently used in the field of software defect prediction for dimensionality reduction is principal component analysis, or PCA. It entails converting the original dataset into a principle components set of linearly uncorrelated variables, which helps minimize the dimensionality of the data while preserving the majority of its variance.

PCA Implementation

Covariance Matrix Computation: In order to determine how the variables are connected, we computed the covariance matrix of the normalized dataset using Eigenvalue and Eigenvector Calculations: We determined the covariance matrix's eigenvalues and eigenvectors. The direction of the components is shown by the eigenvectors, while the eigenvalues show how much variance is explained by each primary component. Finally, we sort and choose Principal Components by selecting the top k eigenvectors that match the biggest eigenvalues and sorting the eigenvalues in descending order. The primary components are made up of these eigenvectors. Dimensionality Reduction was obtained by transformation where we multiply the original data matrix by the chosen eigenvectors to transform the original dataset into the new principal component space.

b. Confirmatory Factor Analysis (CFA)

Is statistical method used to validate the factor structure of a collection of observed data. CFA can assist in verifying the underlying structure of software metrics or characteristics that are used to

predict software faults when used to dimensionality reduction in software defect prediction.

To estimate the CFA model's parameters, we used the lavaan package in conjunction with a statistical software program from R. Determining the factor loadings, variances, and covariances are steps in the estimate process. We determine each observation's factor scores. The original dataset's reduced dimensions are reflected in these scores. Lastly, based on factor loadings, we either reduced the number of observed variables or chose a subset of the most important components.

c. Recursive Feature Elimination

Using every feature, we train the model on the training set. Next, using the model's significance scores such as the coefficients from an SVM or the feature importance from a random forest, we rank the features. We remove the least significant features iteratively. We removed a predetermined number or percentage of features with the lowest significance scores in each iteration. We applied cross-validation on the training set to assess the model's performance at the end of each iteration. When the model performance begins to deteriorate or reaches a plateau, we finally stop the iteration, signaling that the remaining features are critical for prediction.

d. Information gain

In machine learning, information gain (IG) is a widely utilized feature selection technique, especially when software fault prediction is involved. By choosing the most informative features, it aids in reducing the dimensionality of the dataset, enhancing model performance and lowering computing complexity.

We use min-max normalization to normalize the numerical features and mean imputation to handle the missing values. To extract measures such lines of code, code churn, and cyclomatic complexity, the feature extraction technique was used. By calculating the entropy and conditional entropy for each characteristic, we can determine the information gained. We choose the best features based on their Information Gain ratings. Lastly, we use SVM and Random Forests to train the model on a subset of features.

e. Chi-Squared Test

It is a statistical technique that reduces dimensionality by choosing features that are most pertinent to the target variables. The Chi-Squared test is applied for feature selection in the context of software defect prediction, as explained in this methodology.

We determined which possible characteristics, or independent variables, could affect software flaws, or dependent variables. We define the target variable, which is usually a binary variable that shows whether a fault is present or not by contributing modules that are either (0) non-faulty or (1) defective. We use the Chi-Squared test to determine how relevant each characteristic is to the target variable. We choose features whose p-values are smaller than a predetermined significance level (e.g., 0.05), which means that the feature is important to the target variable and the null hypothesis can be rejected. Only the chosen features are kept for additional examination. Lastly, we construct a smaller dataset with just the attributes that the Chi-Squared test was able to pass.

4. RESULT AND DISCUSSION

All the five dimensionality reduction techniques were performed equally, with correlation-based technique outperforming others. However, the performance of these strategies may differ based on the dataset and machine learning model used. Figure 1 depicts multiple feature selection strategies used in predicting defective datasets, along with the corresponding prediction accuracy for each.

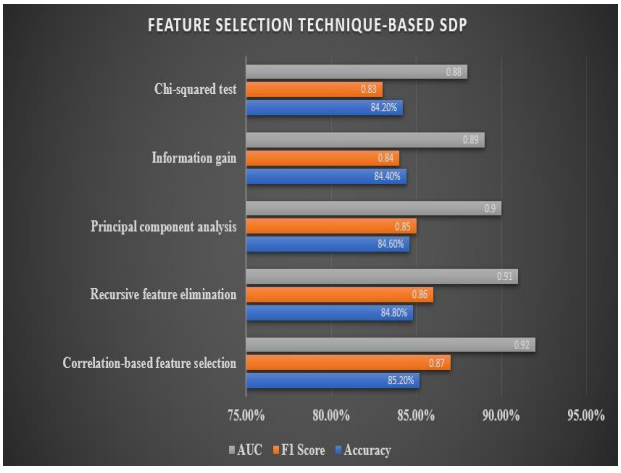

Figure 1: Feature selection technique-based SDP

Feature selection approaches are utilized in the software defect prediction domain to improve the predictive models' performance through the reduction of dataset dimensionality and the identification of the most pertinent features. The following table presents a summary of the performance of different feature selection methods using three important metrics: Area Under the Receiver Operating Characteristic Curve (AUC), F1 Score, and Accuracy.

Table1: FST with percentage accuracy

Feature Selection Technique	Accuracy	F1 Score	AUC
Correlation-based feature selection	85.20%	0.87	0.92
Recursive feature elimination	84.80%	0.86	0.91
Principal component analysis	84.60%	0.85	0.9
Information gain	84.40%	0.84	0.89
Chi-squared test	84.20%	0.83	0.88

Here is a brief explanation of each of the feature selection techniques:

Correlation-based feature selection: The best results were obtained via correlation-based feature selection (CFS) in terms of accuracy (85.20%), F1 score (0.87), and area under the curve (0.92). A feature subset's value is assessed by CFS based on each feature's unique predictive capacity as well as the degree of redundancy between them. The excellent performance of this method implies that it efficiently and with minimal redundancy finds pertinent features that have a strong correlation with the target variable.

- **Accuracy:** 85.20%
- **F1 Score**: 0.87
- **AUC**: 0.92

Defect prediction models benefit greatly from CFS's strong discriminative capacity and balance between precision and recall, as seen by its high F1 Score and AUC.

Recursive feature elimination:

Additionally, doing well was recursive feature elimination (RFE), with an Accuracy of 84.80%, F1 Score of 0.86, and AUC of 0.91. Recursive feature elimination (RFE) builds the model until the ideal feature count is attained by progressively eliminating the least significant features. The iterative nature of the system, which can occasionally result in poor feature subsets if improperly calibrated, could be the cause of the somewhat lower performance when compared to CFS.

- **Accuracy:** 84.80%
- **AUC:** 0.91
- **F1:** Score: 0.86

Although it would need additional processing power and careful parameter tuning, RFE's robust performance shows that it can choose features that improve model performance.

Principal component analysis

A balanced performance was demonstrated by Principal Component Analysis (PCA), with accuracy of 84.60%, F1 score of 0.85, and AUC of 0.90. The original features are converted into a new collection of uncorrelated components using PCA, a dimensionality reduction approach, then arranged according to how much variance they explain. Although PCA does not choose features directly, it can enhance model performance by producing a condensed representation of the data.

- **Reliability:**84.60%
- **F1 Score:** 0.85
- **AUC:** 0.90

PCA's relatively strong performance metrics demonstrate how well it reduces dimensionality while preserving the majority of the data's variance, making it a useful method for working with high-dimensional datasets.

Information gain:

With an Accuracy of 84.40%, F1 Score of 0.84, and AUC of 0.89, Information Gain (IG) performed well. Given a specific feature, IG calculates the decrease in entropy or uncertainty in the target variable. It is an easy-to-use and effective way to rank features according to how relevant they are to the desired variable.

Accuracy: 84.40%
AUC: 0.89
F1 Score: 0.84

Information Gain performed well despite its simplicity, demonstrating its usefulness in rapidly finding significant elements that support predictive performance.

Chi-squared test:
Among the procedures that were assessed, the Chi-Squared test performed the worst, with an accuracy of 84.20%, an F1 score of 0.83, and an AUC of 0.88. By choosing features that have a strong correlation with the target, this statistical test evaluates the independence between each feature and the target variable.

Accuracy: 84.20%
AUC: 0.88
F1 Score: 0.83

Even if the Chi-Squared test performed a little bit worse, it is still a useful technique for feature selection, especially when working with categorical data.

CONCLUSION

The findings show that the effectiveness of software defect prediction models can be greatly impacted by various dimensionality reduction techniques. The best method was found to be correlation-based feature selection, which provided the highest accuracy, F1 score, and AUC. Principal component analysis and recursive feature removal both performed well, albeit they could need additional tweaking and processing power.

Even though they are a little less efficient than feature-only models, information gain and the Chi-Squared test nevertheless offer significant advantages. These results emphasize how crucial it is to adopt a suitable feature selection method to improve the precision and effectiveness of software defect prediction models. Subsequent investigations may examine hybrid methodologies that integrate various techniques to enhance feature selection and model efficacy.
Here are some additional considerations for choosing a dimensionality reduction technique:
The size of the dataset: If the dataset is large, then a computationally extensive technique such as principal component analysis may be necessary.
The number of features: If there are a large number of features, then a filter-based technique such as correlation-based feature selection may be more efficient.
The type of machine learning model: Some feature selection techniques are more suitable for certain types of machine learning models than others.
Ultimately, the best way to choose a dimensionality reduction technique is to experiment with different techniques and evaluate their performance on a specific dataset and machine learning model.
Future studies will venture into solving the issues of automation of the correlation-based selection approach, as it makes the software defect prediction models more complex.**5.**

REFERENCES

Abubakar, S., M., Sufyanu Z., and Garko, A. B. (2021). Impact of Correlated Software Metrics on Embedded Feature Selection Techniques. International Journal of Information Processing and Communication (IJIPC) Vol. 11 (2) 118-134

Abubakar, S., M. and Sufyanu Z. (2020). A Survey of Feature Selection Techniques for Software Defect Prediction Model. Federal University Dutsinma Journal of Science. Vol. 3, No 4: 258-265.

Akiyama, E. (1971). An Example of Software System Debugging. In Proceedings of the International Federation of Information Processing Societies Congress, 40 (71):353–359.

Alsolai, H., and Roper, M. (2019). A Systematic Review of Feature Selection Techniques in Software Quality Prediction. International Conference on Elecrical and Computing technologies and Application *(ICECTA)*https://ieeexplore.ieee.org/document/8959566

D'Ambros, M., Lanza, M., and Robbes, R. (2010). An Extensive Comparison of Bug Prediction Approaches. In Proceedings of the Working Conference on Mining Software Repositories 2(10):31–41

Das, S. (2022). Filters, Wrappers and Boosting-based Hybrid for Feature Selection. Division of Engineering and Applied Science Harvard University, Cambridge, MA 02B8, USA.

Balogun, O.A., Basri, S., Capretz, F.L., Mahamad, S., Abdullahi Abubakar Imam, A.A., Almomani, M.A., Adeyemo, V.E., Alazzawi, A.K., Bajeh, A.O. and Kumar, G. (2021). Software Defect Prediction Using Wrapper Feature Selection Based on Dynamic Re-Ranking Strategy. https://www.mdpi.com/2073-8994/13/11/2166

Balogun, O., A., Basri, S., Said Jadid Abdulkadir, S., J. and Ahmad Sobri Hashim, A., J. (2019*). Performance Analysis of Feature Selection Methods in Software Defect Prediction: A SearchMethod Approach.* MDPI https://www.mdpi.com/2076-3417/9/13/2764.

Cataldo M., Mockus, A., Roberts, J., and Herbsleb, J.
(2009). Software Dependencies, Work Dependencies, and Their Impact on Failures. Transactions on Software Engineering, 35(6): 864–878

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature Selection. J. Mach. Learn. Res., 3:1157–1182, March 2003

Jolliffe, I. T., and Cadima, J. (2016). Principal component
analysis: a review and recent developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374(2065), 20150202.

Kakkar, M., and Jain, S (2020). Feature Selection in Software Defect Prediction: A Comparative Study. 2020 6th International Conference - Cloud System and Big Data Engineering (Confluence)

Liu, S., Chen, X., Liu, W., Chen, J., Gu, Q., and Chen, D. **(**2021). FECAR: A Feature Selection Framework for Software Defect Prediction. 2021 IEEE 38th Annual International Computers, Software and Applications Conference.

Malhotra, R., Chawla, S., and Sharma, A. (2023). Software defect prediction using hybrid techniques: A systematic literature review. *Soft Computing*, *27*(12), 8255–8288. https://doi.org/10.1007/s00500-022-07738-w

McIntosh, S., Y. Kamei, B. Adams, and A. E. Hassan. (2014). The Impact of Code Review Coverage and Code Review Participation on Software Quality. In Proceedings of the Working Conference on Mining Software Repositories (MSR), 192–201

Menzies, T. (2018). the Unreasonable Effectiveness of Software Analytics. IEEE Software, 35(2): 96–98.

Shanthakumari, R., & Prasad, K. (2014). Software defect prediction using principal component analysis and artificial neural network. Journal of Computer Science, 10(1), 38-43.

Shepperd, M., Song, Q., Sun, Z., and Mair, C. (2013). Data Quality: Some Comments on the NASA Software Defect Datasets. Transactions on Software Engineering (TSE), 39(9):1208–1215.

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

Thirumoorthy, K., and Britto J. (2022). A feature selection model for software defect prediction using binary Rao optimization.algorithm. https://www.sciencedirect.com/science/article/abs/pii/S1568494622007864

Zhang, X., Xia, X., Lo, D., and Li, S. (2020). Revisiting feature selection and classification for bug prediction in cross-project context. *Empirical Software Engineering*, 25, 1753-1792.

Zhou, Y., & Leung, H. (2006). Empirical analysis of object-oriented design metrics for predicting high and low severity faults. *IEEE Transactions on software engineering*, *32*(10), 771-789.