Sule Lamido University Journal of Science and Technology (SLUJST) Vol. 2 No. 2 [June, 2021], pp. 33-45

Performance Evaluation of Embedded Feature Selection Techniques over Filter and Wrapper-based Feature Selection Techniques

[1]Shamsuddeen M. Abubakar, [2]Zahraddeen Sufyanu and [3]Ahmed Baita Garko
[1, 2, 3] Department of Computer Science, Faculty of Computing, Federal University Dutse, Ibrahim Aliyu By-pass, P. M.B 7156, Dutse, Jigawa State.
*salsabil012@gmail.com, sufyanzzzzz@gmail.com and abgarko@fud.edu.ng

Abstract

This paper compared the performance of filter-based and wrapper-based feature selection techniques with that of embedded-based feature selection technique. Recursive Feature Elimination (RFE) with SVM Feature selection techniques were explored on Random Forest (F) and Logistic Regression (RFE-LR). Recursive Feature Elimination (RFE) with SVM Feature selection techniques were explored on Random Forest (RF) and Logistic Regression (LR) using ten (10) publicly available defect datasets. SVM-RFE-LR approach of Embedded Feature selection Techniques produced consistent software metrics within the range of 18% to 55% across the datasets, and SVM-RFE-RF reported within 33% to 96%. Significance performance was witnessed over Filter and Wrapper based techniques.

Keywords: *Embedded Feature Selection Techniques, Software Metrics, Defect Prediction*

1. Introduction

One of the fundamental activities in Software Engineering is Software Quality Assurance (SQA). The key role of Software Engineers is to release Defect free software to end users. However, software systems comprise of large and complex correlated software metrics which are found in different software modules. These make construction of software defect prediction model more complicated. Identifying and selecting consistent subset of metrics that improves the performance of software defect prediction model is paramount but challenging problem as it receives little attention in literature.

Software Quality Assurance (SQA) is a process that guarantees all software engineering functional and nonfunctional requirements such as processes, methods, activities and work items are monitored to comply against a defined standard. It consists of all software development processes starting from requirement modeling until release to end users (Abubakar & Sufyanu, 2019). The importance of SQA in distinguishing the top ranked metrics over the other would never be over emphasized. For this reason, project managers find it more essential to outline appropriate software quality improvement plans to mitigate the risk of introducing defects in future releases (Abubakar & Sufyanu, 2019). For example, if code complexity is identified as the top ranked metric, project managers can propose solution to software developers to reduce the complexity of their code, as such reduce the risk of introducing defects before the product is available to end users. These may save industries billions of dollars to find and correct the defective software module after the release (Abubakar & Sufyanu, 2019).

In Software Engineering cycle, testing phase has been considered time and resource consuming, as such difficulties maybe encountered during maintenance plan. Software defect prediction helps in ensuring testing and debugging are made in good time by providing necessary details on the number of defects that are likely to be encountered in a new software release (Lee & Felix, 2017). Feature selection has been widely used as a data preprocessing mechanism for selecting best software metrics before constructing a defect model (Menzies, 2018). It has also been commonly used in software engineering in removing redundant software metrics. According to

Abubakar et al. (2020) feature selection methods are further been divided into three, filter, wrapper and embedded based.

A. Filter Method

Filter-based method were the oldest approaches for feature selection. Filter method algorithms have been used in filtering out irrelevant software metrics that will not be used in the data analysis. This method is usually fast because, they do not constitute learning, but depends on the previous properties of training the data to select and reject features sets (Abubakar et al., 2020). This method is used in selecting software subsets with high number of features and used them as a candidate feature (Noelia & Amparo, 2009).

B. Wrapper Method

The method used machine learning algorithms to selects a subset of metrics as part of its evaluation function. The "black box" function is used in the algorithm to ease in searching of a candidate dataset. Each feature candidate has evaluation function that returns an estimate of the quality of the defect model to be constructed by the learning algorithm. These causes the model to have better prediction accuracy (Abubakar et al., 2020).

C. Embedded Method

In this method, both the machine learning and feature selection are performed together in an embedded form. Embedded methods contain both the characteristics of filter and wrapper-based feature selection methods. It is implementation is usually performed by algorithms that have inbuilt feature selection methods (Abubakar et al., 2020).

It is imperative to stress the need of evaluating the performance of Embedded-based feature selection against Filter and Wrapper-based feature selection techniques. This is because, the former exhibits better prediction accuracy theoretically than the later techniques (Non-embedded). This argument was discussed in a survey of feature selection methods for software defect prediction models, and a Proposed Defect Modelling for Mitigating Correlated Software Metrics, respectively published by the authors of this paper.

Related Literatures

This section presents some related work conducted by various researchers on the field of Software Quality Assurance, consistency and correlation of subset of metrics in particular. Table 1 describes summary of the related literatures.

Table 1: Summary of Related Literatures

S/N	Tittle	Author	Method	Strength	Weakness
1	Metrics Dashboard. A hosted Platform for Software Quality Metrics	Thiruvathukail *et al.,* 2018	Developed a Metrics Dashboard Algorithm	Various metrics can be studied over time and monitor their performance	Consistency and correlation of a metrics was not investigated
2	Performance and cost effectiveness of change burst metrics in predicting software fault	Ndenga *et al.,* 2018	Compared Logistic Regression and Linear Regression Algorithms	Found Burst metrics to have the best numerical performance	Never investigated Defect interpretation
3	AutoSpearman: Automatically Mitigating Correlated Software Metrics for Interpreting Defect Models	Jiarpakdee *et al.,* 2018	Developed Autospearman algorithm	Investigated two Feature Selection Techniques	Embedded based Feature Selection Techniques was not investigated

4	Empirical study of feature selection methods over classification algorithms	Bhalaj *et al.,* 2018	Compares between classifiers of software defect models	Performance of feature selection methods when exposed to different datasets was investigated	correlated software metrics was not investigated
5	Empirical analysis of change metrics for software fault prediction	Choudhary *et al.,* 2018	Compares between change metrics and code metrics	Change metrics with code metrics for defect prediction models	The model performance was not improved
6	The Impact of Correlated Metrics on the Interpretation of Defect Models	Jiarpakdee *et al.,* 2018.	Developed Logistics Regression and Random Forest algorithms	Impact of correlated metrics on the interpretation of defect models was investigated .	Only Filter and wrapper was used in testing results.
7	The Impact of Class Rebalancing Techniques on The Performance and Interpretation of Defect Prediction Models.	Tantithamthavn *et al.,* 2018	Logistics Regression and Random Forest was algorithms was tested	Impact of class rebalancing technique and the interpretation of defect prediction models was investigated	Correlated metrics was not removed before the model construction
8	A Mahalanobis Distance based Algorithm for Assigning Rank to the Predicted Fault Prone Software Module	Chatterjee *et al.,* 2018	Developed Mahalanobis Distance Metrics Algorithm	Increased the prediction accuracy	NASA datasets was tested, which contains many noise in it
9	Filters, Wrapper, and Boosting based Hybrid for Feature Selection	Das 2018	Filters and wrapper-based Feature Selection Techniques was compared	Difference between Filters and Wrapper based was investigated	Embedded based Feature Selection Techniques was not investigated
10	A Framework for Software defect Prediction and Metrics Selection	Huda *et al.,* 2018	Developed a Hybrid Heuristic Algorithm	Consistency of Software Metrics selection was increased	Filter and Wrapper based Feature Selection techniques only
11	Use of Code Similarity Metrics in Software Defect Prediction	Oktan *et al.,* 2018	Used Similarity metrics in developing the algorithm and testing the Correlation	Identifying defective dataset was made easier	Many drawbacks to the method used
12	The Impact of Correlated Metrics on the Interpretation of Defect Models	Jiarpakdee *et al.,* 2019	Developed new Logistics Regression and Random Forest algorithm	Anova datasets analyzed and how to select clean datasets	Improving the consistency of subset metrics receives little attention
13	Proposed Defect Modelling for Mitigating Correlated Software Metrics	Abubakar & Sufyanu 2019	Recursive Feature Elimination with SVM was used	Embedded based feature selection techniques was deployed	Performance evaluation between the three feature selection techniques was not investigated
14	A Survey of Feature Selection Methods for Software Defect Prediction Models	Abubakar *et al.,* 2020	Recursive Feature Elimination with SVM was used	Embedded-based Feature Selection Technique literatures was analyzed	Performance evaluation between the three feature selection techniques was not investigated

Current literatures related to software defects prediction, as concern to finding consistency and correlation of subset of metrics have been reviewed and analyzed. To the best of the authors' knowledge, none of the previous study evaluate the performance of Embedded-based feature selection techniques against Filter and Wrapper-based feature selection techniques.

Current Approach

The methodology of this study is presented here with detailed experimental set-up.

A. Dataset Selection

Initially, 110 publically available datasets were considered. Three evaluation criteria suggested by Jiarpakdee *et al.* (2018) were duly considered in choosing the datasets. The three selection criteria of datasets are explained here:

1. Quality of a datasets is a prerequisite to defect model construction (Shepperd *et al.,* 2013). Therefore, only datasets that are of high quality are considered.

2. Imbalanced data trained using classification techniques tends to favor the majority class.

Therefore, only those datasets which produce non-overly optimistic model performance are used. In case that defective modules tends to be the majority class, than, defect models are most likely to produce overly optimistic performance. Therefore, datasets with defect ratio above 50% are not used.

3. Datasets that can present a precise defect model interpretation are considered. Software analysts regards defect models that are suitable to the datasets, the datasets most presents the value Area under Curve (AUC) of greater than 0.7, (AUC) > 0.7 score, and stable datasets with the value of Event per Variable (EPV) > 10 score.

After carefully studying and identifying the datasets they are minimized to 10 upon application of the above criteria. Table 2 presents the datasets used in conducting the research.

Table 2: The ten (10) datasets with corresponding EPVs and AUCs scores

Project	Datasets	Modules	Metrics	Defective Ratio	EPV	AUC_{LR}	AUC_{RF}
Apache	Xalan 2.6	885	20	46	21	0.79	0.85
Eclipse	Debug 3.4	1,065	17	25	15	0.72	0.81
	JDT	997	15	21	14	0.81	0.82
	Mylyn	1,862	15	13	16	0.78	0.74
	Platform 2	6,729	32	14	30	0.82	0.84
	Platform 2.1	7,888	32	11	27	0.77	0.78
	Platform 3	10,593	32	15	49	0.79	0.81
	SWT 3.4	1,485	17	44	38	0.87	0.97
Proprietary	Prop 1	18,471	20	15	137	0.75	0.79
	Prop 2	23,014	20	11	122	0.71	0.82

B. Feature Selection

The algorithm used is Recursive Feature Elimination (RFE) of feature selection, it is implemented when employing $\sigma 0 < n$ as an input dimensions for the final decision rule. Where $\sigma 0$ is the feature of the sorted list found using greedy backward selection (Abubakar & Sufyanu, 2019).

The algorithm chooses σ0 features iteratively which give the largest separation between the classes, when SVM classifier was used. Such a combinatorial case is solved using a greedy backward method. In each iteration, the training process removes the input dimension which reduces the difference between the input dimensions. Below represents the algorithm of recursive feature elimination and how it operations been perform.

Algorithm 1: Recursive Feature Elimination

1. **Repeat**
2. **Train a SVM resulting in a vector α and scalar β**
3. **Given the solution α, calculate for each feature *p*:**
   $W^2_{(-p)}(\alpha) = \sum \alpha k\, \alpha i y k y l k (X^{-(-p)}{}_k,\ X^{-p}{}_t$
   (where $(X^{-(-p)}{}_k$ means training pont with feature *p* removed).
4. **Remove the feature with smallest value of $\left| W^2(\alpha) - W^2_{(-p)}(\alpha) \right|$**
5. **Until α_0 features remain.**

In step 2, the algorithm is accelerated by removing two or more feature. RFE algorithm presents a better performance on microarray data problems of gene selection (Abubakar *et al.,* 2020). However, the smallest corresponding value of |wi| will be removed by the algorithm in each iteration process (Abubakar & Sufyanu, 2019). In addition, when choosing the backward elimination, using the equation below $\frac{1}{2(n2+n-\sigma o2+\sigma o)}$ in each iteration SVMs has to be trained, this has be avoided when using high input dimension problems are used. The aim of using RFE is to measure and estimate the change in weight W_2 in each iteration by considering only the change in the kernel k from removing one feature, and the vector α remain fixed (Abubakar & Sufyanu, 2019).

To measure the ranking weights for all the features and sorted them in terms of a weight vectors for classification is the purpose of combining SVM and RFE. SVM-RFE is an iteration process of the backward removal of all the features (Abubakar & Sufyanu, 2019).

The process is repeatedly been implemented until only one feature remains in the dataset; the result of the implementation process produce a list of features in the ranking order of their weights vectors. Features with smallest ranking weight are been removed by the algorithm at this stage, while feature variables with significant impact are retained. Finally, the feature variables are listed in descending order of explanatory difference degree. SVM-RFE's selection algorithm can be divided into three stages (Abubakar & Sufyanu, 2019).

Stage 1: classified datasets to be inputted

Stage 2: weight of each feature to be calculated, and

Stage 3: Feature with minimum weight is to be deleted so as to obtain the final ranking of features.

The computational steps of the current approach are described below:

(1) Input Stage

Training sample: $X0 = [X1, X2 \dots \dots \dots . . Xm]T$…….. (a)

Category: $Y = [Y1, Y2 \dots \dots \dots . . Ym]T$ ……... (b)

Set of current feature: $S = [1, 2, \dots \dots \dots n]$…… (c)

Sorted feature list: $r = [\,]$…… (d)

(2) Sorting of Feature Stage

The iteration process is repeated until feature sorted s = [ ] ……. (a)

New training sample is obtained using the matrix according to the remaining features $X = X0\ (S)$ ……..... (b)

The classifier is train using $\alpha = SVM - train(X, Y)$ ……(c)

Weight calculation W = $\sum k\alpha kYkX$k ……. (d)

Sorting standards $Ci = (Wi)2$ …….. (e)

Feature with the minimum weight $f = arg.\min(C)$ …… (f)

Updated sorted feature list $r = [S(f), r]$ ……..(g)

Feature with minimum weight is removed using $S = S(1: -1, f + 1: length(s))$ …… (h)

(3) Output (Feature Sorted List r)

Following are the iteration process

a. The feature with minimum weight $(Wi)2$ will be removed.
b. The SVM then retrains the remaining features to obtain the new feature sorting.
c. SVM-RFE algorithms repeats the process until feature sorted list is produced.
d. On the iteration process, the algorithm manipulates the feature of the sorted list and evaluates the subsets of metrics using the SVM classifier.
e. Finally, optimum feature are obtained.

Figure 1 presents the general method and the steps followed in obtaining the results and the tools used.

RStudio platform was used in conducting the experiments and implementation. The new algorithm is Recursive Feature Elimination (RFE) in conjunction support vector machine with (SVM) was deployed on Random Forest (RF) and Logistic Regression (LR). The packages utilized from RStudio consist of Rnalytica, FSelector, sigFeature, coor and caret with their dependencies. After successfully loading the datasets, then the consistency and correlation of software metrics were recorded in consideration of the following:

i. The embedded feature selection techniques (SVM-RFE-RF and SVM-RFE-LR).

ii. The different training samples of the same datasets for all the datasets on both RFE-RF

with SVM and RFE-LR with SVM.

iii. Comparison of the results against filter and wrapper based feature selections techniques.

RESULTS

The datasets are categorized into four sub classes; each category contains similar metrics as follows:
Category 1 consists of Eclipse 2.0, Eclipse 2.1 and Eclipse 3.0,
Category 2 consists of EclipseJDT and EclipseMylyn,
Category 3 consists of EclipseDebug3.4 and EclipseSWT3.4 and lastly,
Category 4 consists of Prop 1, Prop 2 and Xylan 2.6.

Results from each category were obtained for both SVM-RFE-LR and SVM-RFE-RF approach of Embedded Feature Selection technique so as to measure both consistency and correlation between the two.

Category 1: This category shows the matrices selected by SVM-RFE-LR which includes Eclipse 2.0, Eclipse 2.1 and Eclipse 3.0

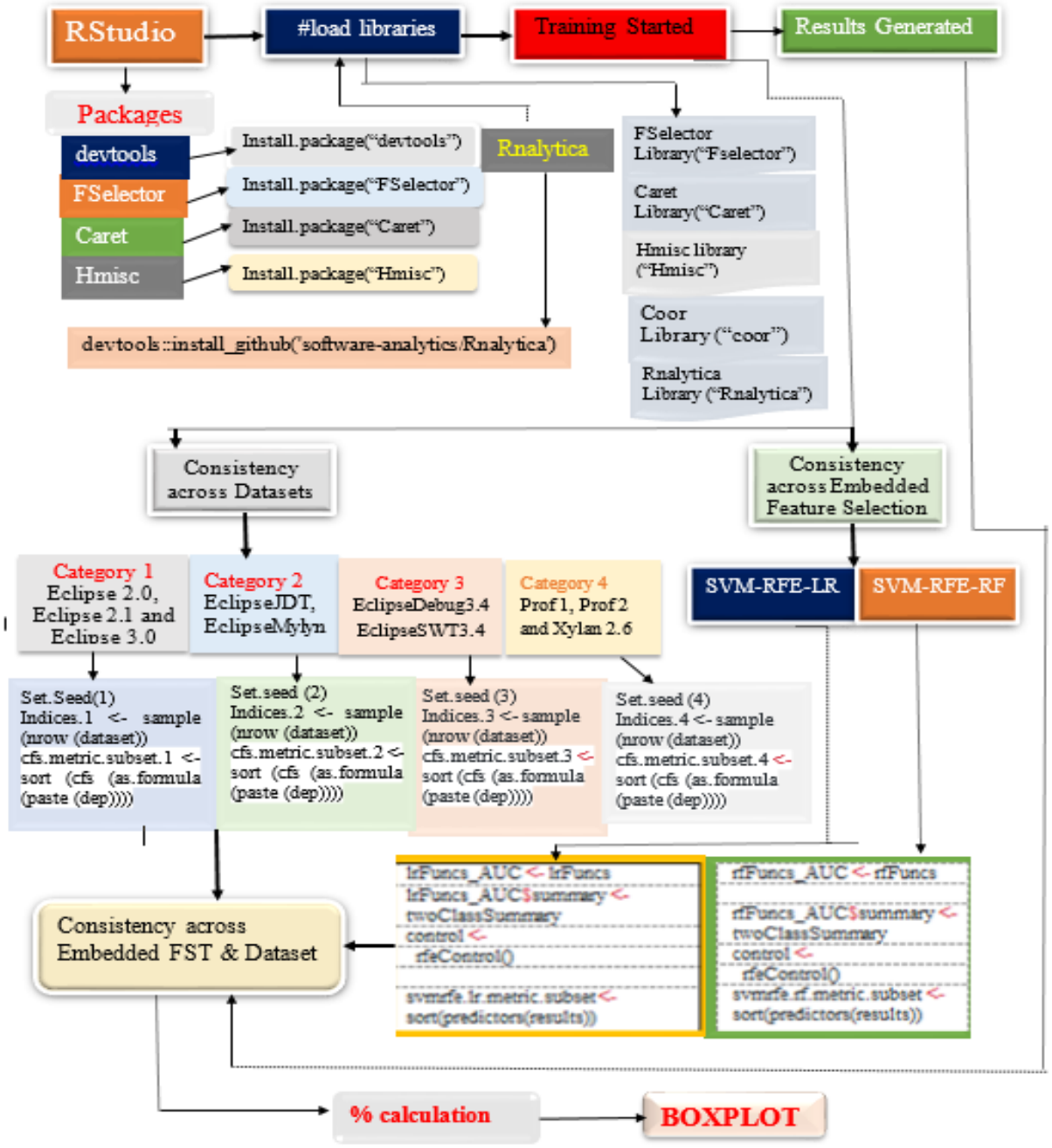

Figure 1: General methodology

Table 3: First Category of Results Generated

Metrics	Eclipse2.0	Eclipse2.1	Eclipse3.0	Intersection	Union
pre	1	1	1	1	1
ADC	1	0	1	0	1
FOUT_avg	1	1	0	0	1
FOUT_max	0	0	0	0	0
FOUT_sum	1	1	0	0	1
MLOC_avg	0	0	0	0	0
MLOC_max	1	0	0	0	1
MLOC_sum	0	0	0	0	0
NBD_avg	1	1	0	0	1
NBD_max	1	0	1	0	1
NBD_sum	1	1	0	0	1
NOF_avg	0	0	0	0	0
NOF_max	1	1	0	0	1
NOF_sum	0	1	0	0	1
NOI	0	0	0	0	0
NOM_avg	1	1	1	1	1
NOM_max	0	1	0	0	1
NOM_sum	0	1	0	0	1
NOTs	0	0	0	0	0
NSF_avg	0	0	0	0	0
NSF_max	0	1	1	0	1
NSF_sum	1	1	1	1	1
NSM_avg	0	1	0	0	1
NSM_max	0	1	0	0	1
NSM_sum	0	1	0	0	1
PAR_avg	1	1	1	1	1
PAR_max	1	1	1	1	1
PAR_sum	1	1	0	0	1
TLOC	0	0	1	0	1
VG_avg	0	0	0	0	0
VG_max	1	1	0	0	1
VG_sum	1	1	1	1	1
Total	16	20	10	6	24

Category 2: These are the datasets which include EclipseJDT and EclipseMylyn. The table presented below shows the represented matrices selected by SVM-RFE-LR.

Table 4: Second Category of Results Generated

Metrics	EclipseJDT	EclipseMylyn	Intersection	Union
NoVU	1	1	1	1
NoFU	0	0	0	0
NoRU	0	0	0	0
NoAU	1	0	0	1
LAU	1	0	0	1
LAU_max	0	0	0	0
LAU_avg	1	0	0	1
LRU	1	0	0	1
LRU_max	1	0	0	1
LRU_avg	1	0	0	1
CCU	0	0	0	0
CCU_max	1	0	0	1
CCU_avg	1	0	0	1
AWRT	1	1	1	1
WAWRT	0	1	0	1
Total	10	3	2	11

Category 3: These are the datasets which include EclipseDebug3.4 and EclipseSWT3.4.

Table 5: First Category of Results Generated

Metrics	EclipseDebug3.4	EclipseSWT3.4	Intersection	Union
LOC	0	1	0	1
AC	0	0	0	0
MC	0	1	0	1
LCOM	1	1	1	1
DIT	0	0	0	0
IFANIN	0	0	0	0
CBO	1	0	0	1
NOC	0	0	0	0
RFC	0	1	0	1
NIM	0	1	0	1
NIV	1	1	1	1
NCM	0	0	0	0
NCV	1	1	1	1
WMC	0	0	0	0
CT	1	0	0	1
Plus	0	0	0	0
Minus	1	0	0	1
Total	6	7	3	10

Category 4: These are the datasets which include Prop1, Prop 2 and Xylan 2.6.

Table 6: First Category of Results Generated

Metrics	Prop1	Prop2	Xylan2.6	intersection	Union
wmc	1	1	1	1	1
dit	1	1	0	0	1
noc	0	1	1	0	1
nbo	1	1	0	0	1
rfc	1	1	0	0	1
lcom	1	0	0	0	1
ca	1	1	0	0	1
ce	1	1	0	0	1
npm	1	1	1	1	1
lcom3	1	1	1	1	1
loc	1	1	1	1	1
dam	1	1	1	1	1
moa	1	1	1	1	1
mfa	1	1	1	1	1
cam	1	1	0	0	1
ic	1	0	0	0	1
cbm	1	1	1	1	1
amc	1	1	1	1	1
max_cc	1	1	1	1	1
avg_cc	1	1	1	1	1
Total	19	18	12	11	20

Consistency of subset of metrics was measured from both embedded techniques using logistic regression and random forest. Table 7 and Table 8 present the results obtained using the embedded-based approach.

Table 7: SVM-RFE-LR Consistency

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

The SVM-RFE-RF produced 33-97% consistent metrics across datasets. The previous research on RFE-RF (non SVM) produced 56.5% consistent metrics at median. Also SVM-RFE-LR produced 18-55% consistent metrics across the datasets. Table 4 presents the summary of results obtained using the two embedded feature selection techniques. Other parameters such as minimum, maximum, first quartile, third quartile and mid-points were all reported.

Table 8: SVM-RFE-RF Consistency

Category	Intersection	Union	Percentage
Category1	30	31	96.77419
Category2	5	15	33.33333
Category3	10	15	66.66667
Category4	9	19	47.36842

Table 9: Data Summary of the Two Embedded Feature Selection

	SVM-RFE-LR	SVM-RFE-RF
N	4	4
Min	18	33
Max	55	96
Sum	158	242
Mean	39.5	60.5
Std. error	9.27811	13.62901
Variance	344.3333	743
Stand. dev	18.55622	27.25803
Median	42.5	56.5
25 prcntil	21	36.5
75 prcntil	55	88.5

Figure 2 presents the boxplot representation of the results obtained using the Embedded-based feature selection techniques.

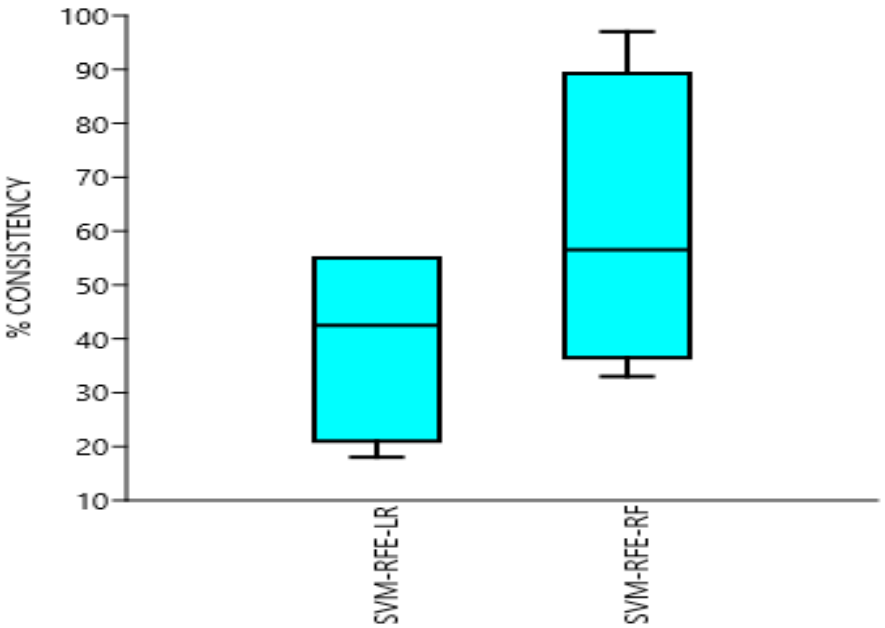

Figure 2: Percentage consistency against two Embedded feature selection techniques

From the boxplot, the minimum observation for the SVM-RFE-LR is 18%, maximum is 55% while the line in the middle is the median which is 42.5% of the observation. It shows a first quartile (Q1) is 21% and third quartile (Q3) is 55%. For SVM-RFE-RF, the minimum observation is 33%, maximum is 96%, and Median of 56.5%. The first quartile is 36.5% while third quartile is 88.5%. Figure 3 presents the comparison of Embedded-based feature selection techniques with non-embedded based (the Filter and Wrapper) based feature selection techniques.

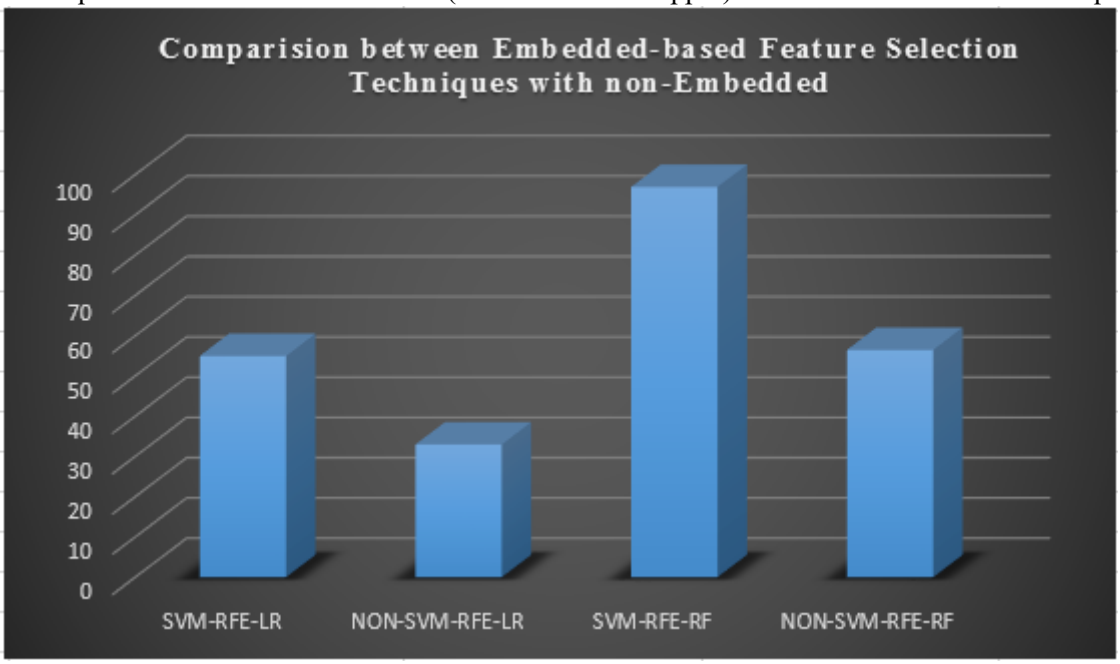

Figure 3: Comparison between Embedded-based and Non-Embedded

From the figure, it can be deduced that, Embedded-based feature selection techniques outperformed their counterparts Filter-based and Wrapper-based feature selection techniques significantly.

Wrapper-based feature selection technique are computationally intensive, which restricts their application to large amount of datasets. However, embedded-based feature selection technique embeds feature selection in the training process of a classifier and are usually specific to some machine learning. They are usually faster than wrapper-based feature selection technique but more likely to overfit.

In case we have large training set, then embedded-based feature selection technique can eventually replace filter-based feature selection technique and wrapper-based feature selection technique.

CONCLUSION

The research was set to compare the performance of Embedded-based feature selection techniques with their counterparts Filter-based and Wrapper-based Feature selection techniques. Following were discovered:

1. SVM-RFE-LR produced 55% consistent metrics as compared to only 33% when Filter-based and Wrapper-based feature selection Techniques were used
2. SVM-RFE-RF produced 97% consistent metrics as compared to 56.5% produced by Filter-based and Wrapper-based feature selection Techniques.
3. The researchers therefore recommend changing the type of datasets on similar work to provide justification of embedded feature selection technique as best feature selection for defect prediction model or otherwise. Besides, allow more datasets exploration and in-depth assessment and rating scales to assess filter, wrapper and embedded feature selection technique for defect prediction models.
4. Finally, the consistency of other types of Embedded feature selection techniques can be investigated.

REFERENCES

Abubakar, S.M., Sufyanu Z., & Abubakar, M.M. (2020). A survey of feature selection
Methods for software defect prediction models. *Federal University Dutse-Ma Journal of Science,* 4(1), 62-68.

Abubakar, S.M., & Sufyanu, Z. (2019). Proposed Defect Modelling for Mitigating Correlated Software Metrics. *Dutse Journal of Pure of and Applied Sciences,* 5(2b), 76-86.

Bhalaji, N., Sundhara, K., & Chithra S. (2018). Empirical Study of Feature Selection Methods over Classification Algorithms. *IEEE Transactions on Software Engineering (TSE),* 2(3):47–80.

Chatterjee S., & Maji, B. (2018). A Mahalanobis Distance Based Algorithm for Assigning Rank To the Predicted Fault Prone Software Modules. IEEE Transaction of Science. *Applied Soft Computing*, 70, 764-772.

Choudhary G. R., Sandep K., Kuldeep K., & Alok M. (2018). Empirical Analysis Of Change Metrics for Software Fault Prediction. *Computer and Electrical Engineering, Elsevier.com/locate/compeleceng,* 67(2): 15-24.

Das Sanmay. (2018). Filters, Wrappers and Boosting-Based Hybrid for Feature Selection. *In Proceedings of the Eighteenth International Conference on Machine Learning (pp.74-81)*

Hoque, Nazrul., Singh, M., & Bhattacharyya, D. K. (2018). EFS-MI: an Ensemble Feature Selection Method for Classification. *Complex & Intelligent Systems,* 4(2), 105-118.

Jiarpakdee J., Tantithamthavorn, C., & Treude, C. (2018, September). Autospearman: Automatically Mitigating Correlated Software Metrics for Interpreting Defect Models. *In 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (pp. 92-103).

Menzies T. (2018). the Unreasonable Effectiveness of Software Analytics. *IEEE Annals of the History of Computing*, (2): 96–98.

Felix, E. A., & Lee, S.P (2017). Integrated Approach to Software Defect Prediction. *IEEE Access, 5, 21524-21547*

Ndega, K. M., Ganchev, I., & Mehat F. (2018). Performance and cost-effectiveness of Change burst metrics in predicting software faults. *Knowledge and Information Systems,* 60(1), 275-302

Okutan Ahmet. (2018). Use of Source Code Similarity Metrics in Software Defect Prediction. arXiv: 1808.10033.

Tantithamthavorn C., Mcintosh S., Hassan E. A., & Kenichi M. (2018). the Impact of Automated Parameter Optimization on Defect Prediction Models. IEEE Transaction on Software Engineering.

Thiruvathukail G.K., Shilpika., Nicholas J. H., & Laufer K. (2018). Metrics Dashboard. A Hosted Platform for Software Quality Metrics. arXiv:1804.02053.