DOI: 10.15439/2023F9248
ISSN 2300-5963 ACSIS, Vol. 37

An Empirical Framework for Software Aging-Related Bug Prediction using Weighted Extreme Learning Machine

Lov Kumar[1], Vikram Singh[2]
Dept. Computer Engineering
National Institute of Technology, Kurukshetra
{lovkumar, viks}@nitkkr.ac.in

Lalita Bhanu Murthy[3]
Dept. CSIS
BITS Pilani Hyderabad Campus
bhanu@hyderabad.bits-pilani.ac.in

Sanjay Misra[4]
Department of Applied Data Science
Institue for Energy Technology, Halden, Norway
ssopam@gmail.com

Aneesh Krishna[5]
School of Elec Eng, Comp and Math Sci (EECMS)
Curtin University Perth, Australia
A.Krishna@curtin.edu.au

***Abstract*—Software ageing (SA) related bugs highlight the issue of software failure within continuously running systems, resulting in a decline in quality, system crashes, resource misuse, and more. To mitigate these bugs, software companies employ various techniques, including code reviews, bug-tracking systems deployment, and thorough testing. Nevertheless, the identification of aging-related bugs remains challenging through these conventional approaches. To address this predicament, early prediction of the affected software regions due to runtime failures can be immensely valuable for software quality assurance teams. By accurately identifying the vulnerable areas, these teams can strategically allocate their limited resources during the testing and maintenance processes. This proactive approach ensures a more efficient and effective bug detection and resolution, enhancing overall software reliability and performance. This study aims to develop aging-related bug prediction models using source code metrics as input. In particular, our objective is to investigate metrics selections, data balancing, and weighted ELM to detect software runtime failure. Experimental results show that ELM with data imbalance SMOTE technique performs the best compared to weighted ELM for addressing the class imbalance problem. The weighted ELM and ELM + SMOTE can predict SA bugs, and these models can be applied to the future releases of software projects for online failure prediction well in advance. The experimental finding shows that the models trained using normal ELM with SMOTE data sampling techniques have significant performance improvement.**

***Index Terms*—Functional Requirements, Non-Functional Requirements, Data Imbalance Methods, Feature Selection, Classification Techniques, Software Aging**

I. Introduction

IN TODAY'S scenario, software (SW) companies are adopting Object-Oriented (OO) concepts to develop modern software systems. The primary reason for adopting these concepts is due to their efficient functionalities, like reusability (extending the code use again), inheritance, data abstraction, polymorphism, cohesion, and coupling. These functionalities help to design SW with high quality such as maintainability, reliability, portability, and reusability. Estimating the SW quality and finding its correlation with static code metrics can help testers, architects, and requirement analysts to analyze the source code concerning SW quality before deploying[1][2]. This point is our primary motivation for present work, with an aim to find the correlation between Software Aging (SA) related bugs and static code metrics as both cost and effort to fix run-time failures or Aging-related bugs increase exponentially if the reason for these failures is not identified prior to SW deployment [3] [4]. In such contexts, the utilization of software aging prediction models emerges as a valuable asset during the initial stages of the SW development life cycle (SDLC). Furthermore, their application holds the potential to enhance software quality, diminish testing expenses, and streamline maintenance efforts.

Software development companies often intend to consider different techniques, such as code reviews, deployment of bug-tracking systems, and various testing techniques for reducing SA bugs [3] [4]. However, it is quite an arduous task to discover the region of SW to be affected due to runtime failure [5]. This study aims to address the issue of predicting runtime failures related to SA by developing a model that incorporates source code metrics and aging-related data. By combining these elements, we strive to create a robust prediction model capable of identifying potential runtime failures in SW systems.

More specifically, this work aims to study the relationship between various source code metrics and SW aging-related bugs and develop a machine learning (ML) enabled prediction model to proactively predict these bugs. These ML models would steer the identification of patterns within the future versions of the SW system based on source code metrics for bug detection. However, we found two major issues in the development of aging-related bugs[6][7][8][9]:

- ***High-Dimensional Data:*** Before applying any technique for model development, it is essential to select relevant and right sets of features that are significant for the development of ML models. In this study, we have considered five different feature ranking techniques: Gain Ratio (GR), OneR, Relief-F(RF), Information Gain (IG), and Symmetric Uncertainty (SU) for ranking and selecting the significantly relevant features for the development of bug prediction models [6][7].

- ***Imbalanced Data:*** Developing an effective prediction model becomes very challenging, particularly with training over highly imbalanced data, it is another pertinent issue for designing SA-related bug prediction models. In these settings, ELM has emerged as a highly efficient and effective ML technique with interest across various domains in recent years. ELM utilizes least square learning methods within a single hidden layer neural network, forming the foundation of its concept for the creation of robust regression and classification models. In this study, the Weighted ELM (WELM) technique has been considered to handle the imbalanced data and develop an effective model for SW aging prediction [8][9]. Further, the performance of these models developed using WELM has been compared with the data imbalance technique and unweighted ELM [8][9].

This work focuses on conducting extensive experimentation to design intelligent models that utilize machine-learning techniques for the proactive prediction of ageing-related bugs in SW. To achieve this, various approaches were employed, including the selection of effective metrics, data balancing, and pattern identification using weighted ELM. Additionally, the data balancing techniques employed addressed class imbalance issues, ensuring that the models were trained on a well-represented dataset. By leveraging these techniques, this study aimed to enhance the proactive identification of SW ageing-related bugs, enabling developers to mitigate potential issues and improve overall SW reliability. Accordingly, the respective Research Questions (RQs) are justified in this study:

RQ1: *What benefits on the performance of aging prediction models after removing ineffective metrics?*

RQ2: *What benefits on the performance of aging prediction models after changing kernel functions?*

RQ3: *What is the benefit of using weighted ELM over ELM + SMOTE techniques for aging prediction models?*

The rest of the paper is organized as follows: Literature Review on methods used for SA bugs is presented in SectionII. The solution methodology, experimental datasets, as well as the various performance parameters used to compare the developed models, are described in SectionIII. The experimental finding of this work and comparative analysis of models developed using different methods are described in SectionIV. Finally, Section VIIwraps up the material presented and suggests research directions for future studies.

II. Related work

In today's scenario, the SW systems operate continuously to complete the assigned task. However, due to faults in design, development, testing, and inappropriate application environment, there is a chance of occurrence of bugs during run time that eventually causes software ageing (SA). Here, we present some key background concepts related to SA.

The idea of SA was first introduced by Huang et al. [10] and subsequently, other researchers have extended it in order to recognize this significant phenomenon [3][11]. Specifically, Parnas et al. [3] discussed the reason behind SA and characterized two types of SA: first, the malfunction of the manufactured goods prior to transforming it to gather transforming needs and second, the effect of the modifications that are prepared. Similarly, Alonso et al. [11] have performed a comparative study related to software rejuvenation and have demonstrated SA in 6 different ways from ground to granularity level. Further, Matias [4] focused on highlighting the potentially common problems that occur due to SA presence, such as data discrepancy, statistical errors, and exhaustion of operating system (OS) resources, which are sample demonstrations of SA. There are a good number of classes where SA effects are reported in the literature with associated impacts to running down of OS resources, and predominantly those connected to the functioning of main memory [12][13].

Zheng and his group developed different rejuvenation rules to find time-based constraints[14]. They have conducted a systematic study to measure these rules numerically and observed that they are better than Markovian arrival processes (MAPs). They observed that eliminating all bugs is not practically possible. Therefore, efforts are being made to estimate the run-time failure or SA of an SW system with the objective of avoiding future system failures. The process of identifying and predicting these bugs is a challenging task for software engineers. Padhy et al. [15] proposed an aging prediction system based on re-usability optimization. This prediction system estimates the re-usability level of the software components. They have applied the concept of re-usability risk management and found that aging-related systems are excessively reused systems. The other method to avoid aging failures is Rejuvenation. Sharma and Kumar have used different types of ensemble models to develop SA prediction models [16]. They have validated the effectiveness of ensemble learning for SA prediction using LINUX and MYSQL bug datasets. They have observed that ensemble methods can identify bugs at early stages and can help reduce the cost and damage caused due to SA.

Khanna and her team have used Artificial Immune Systems for developing SA bugs prediction classifiers [17]. They have used five different types of open-source SW systems to validate the proposed models and asserted that these models have the ability to predict SA bugs. Similarly, Fangyun Qin and his team [18] have examined the variation performance of the models for cross-project SA bugs prediction using different normalization methods, kernel functions, and ML-based classifiers. They have adopted the Scott-Knott test technique to validate their finding and observed that the performance of cross-project SA bugs prediction models depends on the classifiers and kernel functions, while normalization methods do not impact much on performance.

From the above studies and developments, we observed that many researchers have addressed the problem of SA bug prediction. However, the prediction models trained on imbalanced data have rarely been addressed in the literature. Thus, this research work will be a pioneer in the development of SA bugs on imbalanced datasets. In this work, we empirically investigate the performance of SA prediction models developed using weighted ELM and ELM with separate methods for data sampling i.e., SMOTE.

III. Methodology

This section presents the methodology adopted in this experiment in order to predict SA using various ML techniques. Figure 1 illustrates the proposed framework for the development of the SA bug prediction model considering publicly available datasets from seven large open-source software systems. In the discussion in the previous section, it is observed that there have been different types of static code metrics used for developing intelligent models to detect SA bugs[19][20] [21][22]. Therefore, we have applied different sets of static code metrics, such as McCabe's cyclomatic complexity, Halstead's set of metrics, metrics related to the size of software, and metrics associated with aging bugs for aging-related bug prediction. Since we are using these sets of metrics as input, so it is compulsory to remove ineffective metrics, which may help improve the models' performance.

The proposed solution's first phase is applying the feature selection concept to remove ineffective metrics and compute the best sets of effective metrics on pre-processed datasets. Here, we have used five techniques to remove ineffective features such as Gain Ratio, Symmetric Uncertainty, OneR, RELIEF, and Information Gain. The concepts of these techniques are based on performance parameters to rank the features and select top-ranked features for the analysis. In this work, we have used $\lceil log_2^n \rceil$ numbers of top features as the best sets of effective features i.e., $\lceil log_2^{82} \rceil$=7.

The next phase of the proposed framework involves balancing data using SMOTE techniques. We have used ELM with SMOTE and WELM to find patterns to predict SA-related bugs. Here, WELM is applied separately because it was observed that the WELM could handle class imbalance problems. Finally, the ability of the model prediction trained by using ELM with SMOTE and WELM is computed in terms of AUC and Accuracy. This research framework uses two projects, Linus and MySQL, which are downloaded from the PROMISE data repository. We have considered four variants of Linus and three variants of MySQL. Table I presents the description of Linux and MySQL project with the total number of classes(Total Classes), number of non-ageing classes(Non-Aging), number of Aging classes(Aging), % of non-ageing classes(% Non-aging), and % of ageing classes (%Aging).

TABLE I: Software Projects Description

	Linux				MySQL		
Name	**Driver Net**	**Ext3**	**Driver Scsi**	**Ipv4**	**Optimizer**	**Replication**	**Innodb**
ID	**Proj1**	**Proj2**	**Proj3**	**Proj4**	**Proj5**	**Proj6**	**Proj7**
Total Classes	2292	29	962	117	36	32	402
Non-Aging	2283	24	958	115	33	28	370
Aging	9	5	4	2	3	4	32
%Non-Aging	99.61	82.76	99.58	98.29	91.67	87.5	92.04
%Aging	0.39	17.24	0.42	1.71	8.33	12.5	7.96

Feature Ranking: In this work, we have considered 82 different source code metrics as input to the models used for SA prediction. In order to achieve better performance, five different feature ranking techniques are used for feature selection. In this method, a performance parameter is used to rank the features, and the top $log_2 n$ features out of n number of features are selected for aging prediction. The output of two feature ranking techniques i.e., Gain ratio and relief are shown in TableII. Table II presents the ranking of software metrics using two feature ranking techniques from rank 1 to 82 metrics for all 3 projects.

Similarly, the rank of software metrics is computed using other three feature ranking techniques i.e., oneR, infogain,

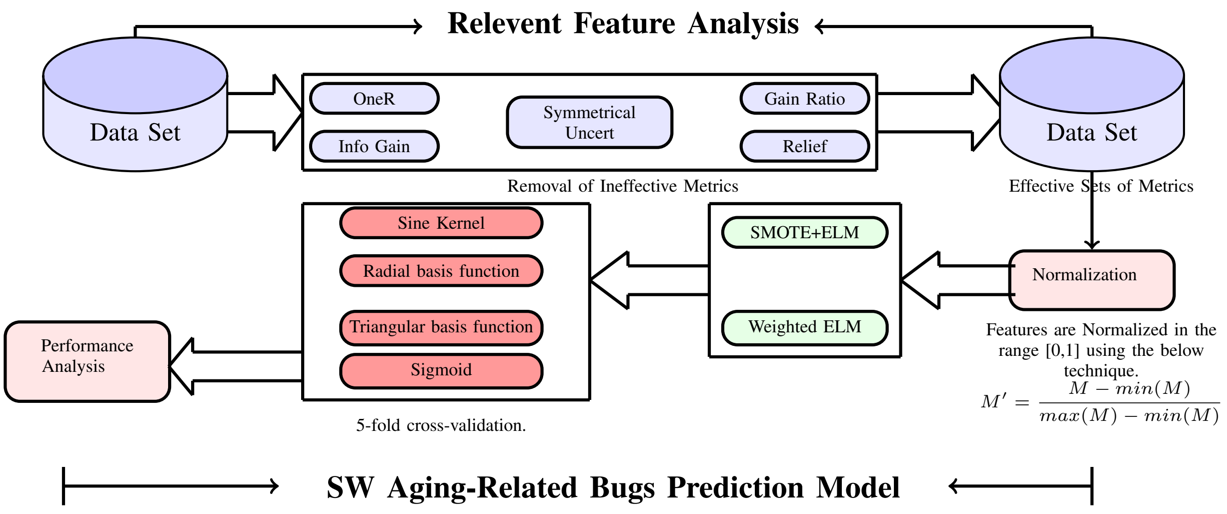
Relevent Feature Analysis
Data Set
OneR
Info Gain
Symmetrical Uncert
Gain Ratio
Relief
Data Set
Removal of Ineffective Metrics
Effective Sets of Metrics
Sine Kernel
Radial basis function
Triangular basis function
Sigmoid
SMOTE+ELM
Weighted ELM
Normalization
Performance Analysis
Features are Normalized in the range [0,1] using the below technique.
$M' = \frac{M - min(M)}{max(M) - min(M)}$
5-fold cross-validation.
SW Aging-Related Bugs Prediction Model

Fig. 1: Conceptual Scheme of Proposed Framework

Symmetric Uncertainty (SU) ranking techniques. It is observed from Table II that the rank of AltAvgLineBlank metrics is 38, 82, and 82 for projects Proj1, Proj2, and Proj3 respectively when gain-ratio feature ranking technique is applied.

IV. Experimental Results

RQ1: What benefits on the performance of aging prediction models after removing ineffective metrics? The AUC value is computed for every model over multiple cutoff points, these values of AUC curve along with Accuracy and F-Measure are listed in Tables III and IV. Here, the results are presented over different projects, different classification techniques, and one data imbalance technique. The feature ranking technique yielding the high AUC for a given project is depicted in green color. Further, inference from Tables III and IV, the AUC value of proj1 using oneR with the sigmoid kernel is better than the other feature ranking techniques and kernel methods. In most of the cases, the performance parameters values of the feature ranking techniques with ELM + SMOTE were found to be comparable or even better than feature ranking techniques with weighted ELM.

Comparison of Feature Ranking Techniques using Boxplots and Descriptive Statistics: Figures 2 and 3 depict boxplots for comparing the minimum, maximum, interquartile range, degree of dispersion, and outliers in the AUC, F-Measure, and Accuracy for all feature ranking techniques. Table refdst reports the descriptive statistics of the performance of different feature ranking techniques. From Figures 2 and 3, we infer that the accuracy and F-measure values of the case where feature ranking techniques are used with ELM + SMOTE is better than the one in which we use feature ranking techniques with weighted ELM. However, these performance parameters such as accuracy and F-measure are not those good parameters to validate the model developed using imbalanced data sets. So, in this experiment, Area under the ROC (Receiver Operating Characteristics) Curve (AUC) values have been considered to measure classifier performance. The line is represented using the red color of the Figures 2 and 3 displays the median points of the data and this line is used to divide the box into two segments. Similarly, Figures 2 and 3 depict the average AUC of RF in the case of ELM + SMOTE is higher than the corresponding values for other feature ranking techniques.

Comparison: Null Hypothesis: Feature Ranking Techniques: Statistical Significance Testing: After comparing different feature ranking techniques using boxplots and Descriptive Statistics, Wilcoxon signed-rank test with a Bonferroni correction has been applied for statistical hypothesis testing. The objective of this testing is to investigate the statistical difference between the pairs of different feature ranking techniques.

The results of the Wilcoxon signed-rank test with a Bonferroni correction of the feature ranking technique are shown in Figures 4 and 5. Figures 4 and 5 consist of a green and red dots. The rejected null hypothesis is represented using a red dot and the accepted null hypothesis is represented using

TABLE II: Ranking of Static Code Metrics for all Projects using Gain Ratio and Relief.

	Gain Ratio			Relief		
	Proj1	Proj2	Proj3	Proj1	Proj2	Proj3
CountDeclInstanceVariable	43	10	17	52	61	81
CountLineInactive	20	42	47	32	51	36
CountClassDerived	46	36	40	57	57	67
CountLineBlank	2	17	48	10	22	22
AvgCyclomaticStrict	53	31	33	50	13	47
CountDeclMethodPrivate	82	8	21	80	81	59
MaxCyclomaticModified	64	78	75	45	6	33
CountDeclClass	42	35	38	56	56	71
MaxCyclomatic	63	80	77	44	4	34
n2	23	44	10	3	37	4
N1	4	45	56	9	42	8
CountDeclMethodConst	74	5	23	78	79	70
CountLineCodeExe	7	15	71	26	38	2
MinEssentialKnots	76	64	53	71	71	52
CyclomaticStrict	62	73	80	74	60	64
DeallocOps	34	55	60	31	41	37
RatioCommentToCode	72	62	51	47	44	82
CountDeclInstanceVariablePublic	58	2	19	75	82	73
Dif	13	61	11	7	5	26
DerefUse	21	57	59	21	12	15
MaxInheritanceTree	67	76	64	62	64	57
CountStmtEmpty	70	67	67	51	46	50
CountDeclMethodProtected	77	12	25	81	76	51
CountStmtDecl	17	65	8	23	20	25
CountDeclFunction	26	13	26	6	8	28
CountDeclClassMethod	47	33	28	53	55	77
AllocOps	35	58	61	37	40	38
CountDeclMethod	71	6	16	82	77	69
MaxCyclomaticStrict	65	75	65	46	3	32
AvgLineCode	55	40	44	34	25	40
CountLineCode	31	14	66	20	35	10
CountStmtExe	24	66	74	16	9	5
SumCyclomatic	16	51	50	38	19	13
CountPath	79	71	68	66	65	61
CountDeclMethodPublic	78	18	20	76	73	56
MaxNesting	66	74	54	63	63	53
Essential	61	79	79	60	69	63
n1	10	47	4	2	2	24
DerefSet	11	23	1	19	28	20
CountClassCoupled	49	34	27	55	58	66
CountLine	27	19	41	4	32	16
CountClassBase	48	37	39	58	53	65
AvgCyclomaticModified	41	25	36	49	16	45
AvgEssential	54	28	37	41	26	49
AltCountLineComment	15	24	31	25	39	31
SumCyclomaticModified	12	50	55	39	15	19
PercentLackOfCohesion	57	63	52	72	59	79
AltAvgLineComment	40	1	34	35	23	39
AvgCyclomatic	52	26	30	48	7	48
UniqueDerefUse	32	56	2	14	18	23
CountLineComment	14	41	12	27	36	30
CountDeclInstanceMethod	44	21	18	59	52	80
AltAvgLineBlank	38	82	82	29	30	44
Cyclomatic	59	72	76	69	68	76
CountStmt	25	68	73	18	11	12
MaxEssentialKnots	68	77	57	65	66	55
AltAvgLineCode	51	30	35	33	27	41
AltCountLineBlank	1	29	32	8	33	21
SumCyclomaticStrict	19	49	49	42	10	9
Knots	60	81	78	70	70	58
Vol	9	59	63	13	48	1
CountDeclInstanceVariableProtected	50	7	13	67	75	74
CountOutput	80	46	69	64	62	60
AvgLineBlank	37	38	45	28	29	46
SumEssential	28	48	3	15	1	3
CountDeclMethodAll	75	3	22	77	78	68
AltCountLineCode	33	27	29	17	34	17
CountDeclInstanceVariablePrivate	56	9	14	61	67	75
CountInput	81	20	46	73	72	54
AvgLine	30	32	43	30	14	43
Eff	8	60	62	24	50	18
AvgLineComment	39	39	42	36	24	42
N2	5	52	9	12	45	6
Len	3	53	58	11	43	7
Voc	18	54	7	1	31	14
CountLinePreprocessor	36	43	70	43	47	35
CountLineCodeDecl	29	16	72	40	49	27
CountDeclClassVariable	45	22	24	54	54	78
CountDeclMethodFriend	73	11	15	79	80	72
CountSemicolon	22	70	6	22	21	11
CyclomaticModified	69	69	81	68	74	62
UniqueDerefSet	6	4	5	5	17	29

TABLE III: AUC: Accuracy: Weighted ELM

	AUC																			
	Sine					Radial basis function					Triangular basis function					Sigmoid				
	OneR	IG	GR	RF	SU	OneR	IG	GR	RF	SU	OneR	IG	GR	RF	SU	OneR	IG	GR	RF	SU
Proj1	0.63	0.60	0.60	0.58	0.43	0.58	0.50	0.50	0.50	0.50	0.48	0.50	0.50	0.50	0.50	0.8	0.70	0.66	0.72	0.67
Proj2	0.9	0.41	0.79	0.61	0.80	0.79	0.88	0.56	0.59	0.57	0.57	0.49	0.57	0.50	0.47	0.70	0.71	0.85	0.80	0.82
Proj3	0.67	0.61	0.55	0.53	0.51	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.67	0.71	0.58	0.69
Proj4	0.37	0.86	0.65	0.51	0.62	0.68	0.39	0.47	0.50	0.45	0.49	0.45	0.48	0.50	0.46	0.58	0.56	0.81	0.73	0.80
Proj5	0.59	0.44	0.47	0.57	0.48	0.50	0.50	0.49	0.50	0.49	0.52	0.50	0.51	0.50	0.49	0.76	0.67	0.74	0.66	0.71
Proj6	0.35	0.21	0.26	0.64	0.55	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.56	0.59	0.56	0.55	0.55
Proj7	0.54	0.55	0.70	0.46	0.54	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.77	0.86	0.70	0.61	0.77
	Accuracy																			
	Sine					Radial basis function					Triangular basis function					Sigmoid				
	OneR	IG	GR	RF	SU	OneR	IG	GR	RF	SU	OneR	IG	GR	RF	SU	OneR	IG	GR	RF	SU
Proj1	71.20	52.66	52.66	49.61	52.14	94.28	99.61	99.61	99.61	99.61	95.99	99.61	99.61	99.61	99.61	59.86	51.27	43.06	43.98	34.42
Proj2	80.56	81.81	83.26	71.52	84.82	83.06	75.99	87.32	92.41	88.15	89.71	98.13	89.19	98.96	93.04	65.38	67.15	70.06	60.19	63.93
Proj3	58.62	62.07	37.93	62.07	44.83	82.76	82.76	82.76	82.76	82.76	82.76	82.76	82.76	82.76	82.76	31.03	58.62	51.72	31.03	48.28
Proj4	72.65	72.65	78.63	52.14	73.50	85.47	76.07	92.31	98.29	88.89	95.73	88.03	94.87	98.29	90.60	64.96	61.54	63.25	47.86	61.54
Proj5	77.86	47.26	50.00	67.66	51.99	92.04	92.04	91.04	91.54	85.82	92.29	91.54	92.04	91.29	91.04	62.94	62.69	62.94	63.93	64.18
Proj6	36.11	38.89	47.22	61.11	72.22	91.67	91.67	91.67	91.67	91.67	91.67	91.67	91.67	91.67	91.67	19.44	25.00	19.44	16.67	16.67
Proj7	56.25	59.38	65.63	62.50	56.25	87.5	87.5	87.5	87.5	87.5	87.5	87.5	87.5	87.5	87.5	78.13	75.00	65.63	50.00	59.38

TABLE IV: AUC: Accuracy: ELM + SMOTE

	AUC																			
	Linear					Polynomial					Sigmoid					Radial basis function				
	OneR	IG	GR	RF	SU	OneR	IG	GR	RF	SU	OneR	IG	GR	RF	SU	OneR	IG	GR	RF	SU
Proj1	0.50	0.49	0.48	0.50	0.48	0.57	0.62	0.61	0.72	0.62	0.57	0.50	0.50	0.50	0.50	0.72	0.50	0.50	0.50	0.50
Proj2	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.59	0.68	0.69	0.55	0.65
Proj3	0.89	0.75	0.76	0.89	0.78	0.68	0.91	0.80	0.81	0.81	0.40	0.50	0.50	0.56	0.50	0.50	0.60	0.65	0.50	0.60
Proj4	0.50	0.50	0.50	0.48	0.50	0.49	0.50	0.49	0.83	0.49	0.50	0.50	0.50	0.50	0.50	0.48	0.48	0.49	0.50	0.48
Proj5	0.85	0.76	0.77	0.66	0.78	0.79	0.80	0.79	0.85	0.83	0.73	0.50	0.50	0.81	0.50	0.77	0.63	0.64	0.88	0.65
Proj6	0.66	0.96	0.96	0.88	0.84	0.63	0.65	0.60	0.66	0.51	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
Proj7	0.92	0.85	0.85	0.94	0.98	0.92	0.85	0.88	0.87	0.82	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
	Accuracy																			
	Linear					Polynomial					Sigmoid					Radial basis function				
	OneR	IG	GR	RF	SU	OneR	IG	GR	RF	SU	OneR	IG	GR	RF	SU	OneR	IG	GR	RF	SU
Proj1	98.44	97.30	95.05	98.30	95.13	98.57	98.39	76.40	95.60	78.63	93.31	98.43	98.39	98.43	98.43	98.22	98.39	98.44	98.43	98.43
Proj2	98.24	98.33	98.33	98.25	98.33	98.34	98.33	98.33	98.35	98.33	98.34	98.33	98.02	98.35	98.33	98.34	98.23	98.54	96.80	98.23
Proj3	89.19	77.78	77.78	89.19	80.56	67.57	91.67	80.56	81.08	80.56	40.54	55.56	55.56	56.76	55.56	48.65	55.56	61.11	48.65	55.56
Proj4	96.52	96.52	96.52	92.24	96.52	94.78	95.65	94.78	89.66	94.78	96.52	96.52	96.52	96.55	96.52	93.04	93.04	94.78	95.69	93.04
Proj5	87.58	80.70	79.33	64.73	78.68	85.33	86.62	86.44	89.51	88.13	72.46	71.93	71.56	75.45	71.87	85.33	79.39	79.33	92.19	79.78
Proj6	48.65	94.59	94.59	86.84	86.49	67.57	70.27	72.97	60.53	59.46	75.68	75.68	75.68	76.32	75.68	75.68	75.68	75.68	76.32	75.68
Proj7	90.48	83.33	80.95	92.86	97.44	90.48	80.95	85.71	83.33	76.92	61.90	61.90	61.90	61.90	64.10	61.90	61.90	61.90	61.90	64.10

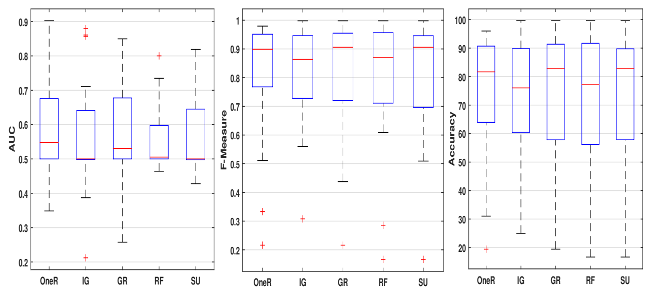

Fig. 2: Box-and-Whisker plot: AUC Value of Weighted ELM with Feature ranking techniques

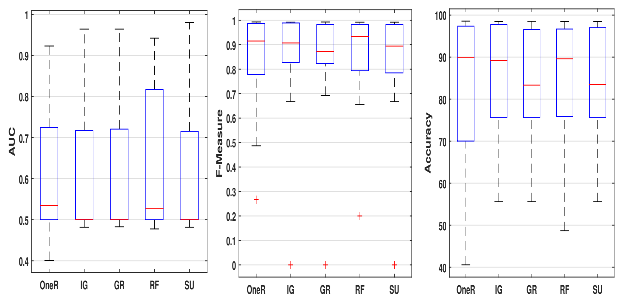

Fig. 3: Box-and-Whisker plot: AUC Value of ELM + SMOTE with Feature ranking techniques

a green dot. The null hypothesis in this experiment is that "there is no statistically significant difference between the two techniques". In this experiment, the standard cut-off value of $0.05/(\textit{total number of unique pairs})$=0.05/10=0.005 is used to reject and accept this hypothesis. The results shown in the Figures 4 and 5 depict that all cells contain a green dot for feature ranking techniques in both weighted ELM and ELM + SMOTE cases. Based on these results, it is observed that the aging prediction model developed by considering different sets of metrics obtained using feature ranking techniques is not significantly different.

RQ2: What benefits on the performance of aging prediction models after changing kernel functions?

After finding relevant sets of features using five different feature ranking techniques, the aging prediction models are developed using weighted ELM with various kernels and ELM with SMOTE data imbalance technique. In this study, four different types of kernel functions are used to develop a model

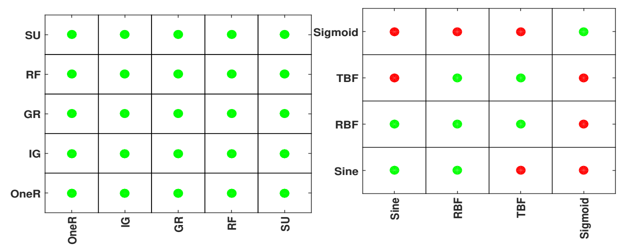

Fig. 4: Weighted ELM: Wilcoxon Signed-Rank Test + Bonferroni Correction: A Red Dot Means that H_0 is Rejected

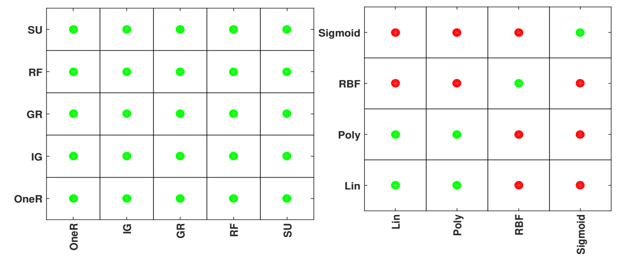

Fig. 5: ELM + SMOTE: Wilcoxon Signed-Rank Test + Bonferroni Correction: A Red Dot Means that H_0 is Rejected

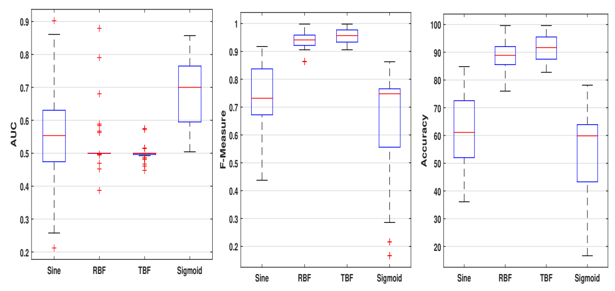

Fig. 6: box-and-whisker plot: AUC Value of Weighted ELM with different kernels

for predicting SA bugs. The developed models are validated using 5-fold cross-validation. The value of the AUC curve along with accuracy and F-Measure for different kernels are shown in Tables III and IV. The results are presented over different projects, different feature ranking techniques, andone data imbalance technique. From the Tables III and IV, it can be inferred that AUC values of models using sine and sigmoid kernels are better than the other radial basis function and triangular basis function kernels in the case of weighted ELM. Similarly, the AUC value of linear, polynomial, and radial basis function kernels are better than the sigmoid kernel in the case of ELM + SMOTE

Comparison of kernel functions using Boxplots and Descriptive Statistics: Figures 6 and 7 depict boxplots for comparing the minimum, maximum, interquartile range, degree of dispersion, and outliers in the AUC, F-Measure, and Accuracy for all considered kernel functions. From Figures 6 and 7, we infer that the accuracy and F-measure values of the case where kernel functions are used with ELM + SMOTE

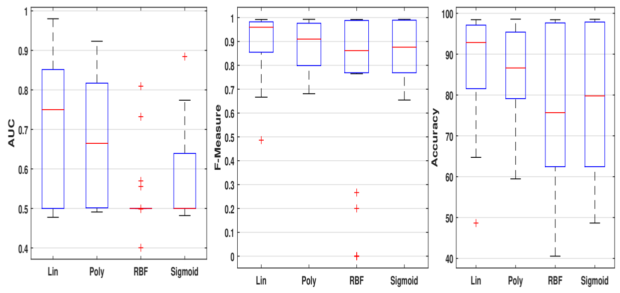

Fig. 7: box-and-whisker plot: AUC Value of ELM + SMOTE with different kernels

are better than the one in which we use kernel functions with weighted ELM. In this experiment, Area under the ROC Curve (AUC) values has been considered to measure classifier performance. Figures 6 and 7 depict the average AUC of sigmoid kernel function in case of weighted ELM is higher than the corresponding values for other kernel functions.

Comparison: Null Hypothesis: kernel functions: Hypothesis Statistical Significance Testing: The results of the Wilcoxon signed-rank test with a Bonferroni correction of the different pairs of kernels are shown in Figures 4 and 5. The rejected null hypothesis is represented using a red dot and the accepted null hypothesis is represented using the green dot. In this experiment, a standard cut-off value of $0.05/(\textit{total number of unique pairs})=0.05/6$ has been considered to reject and accept this hypothesis. The results shown in Figures 4 and 5 depict that the aging prediction models developed using weighted ELM with sine and RBF kernel function are not significantly different. Similarly, the SA prediction models developed using weighted ELM with sine, TBF, and sigmoid kernel functions are significantly different.

RQ3: What is the benefit of using weighted ELM over ELM + SMOTE techniques for aging prediction models? Zong et al.[8] mathematically proved that weighted ELM with various kernel functions is able to handle imbalanced data and also maintain better performance on balanced data as compared to unweighted ELM. In this work, we have considered weighted ELM and unweighted ELM with data imbalance techniques i.e., SMOTE (Synthetic Minority Oversampling Technique) to develop a SA bugs prediction model. SMOTE technique is based on the oversampling concept and aimed to increase the number of artificial instances that belong to the minority class. Specifically, artificial instances are created using oversampling.
The value of the AUC curve along with Accuracy and F-Measure for WELM and ELM + SMOTE listed in Tables III and IV. According to Tables III and IV, the performance parameters i.e., AUC, Accuracy, and F-Measure of the scenario where weighted ELM is used are lower or comparable with that of the one in which ELM with data imbalance SMOTE technique is used.

Comparison of weighted ELM and ELM + SMOTE using Boxplots and Descriptive Statistics: Figure 8 presents

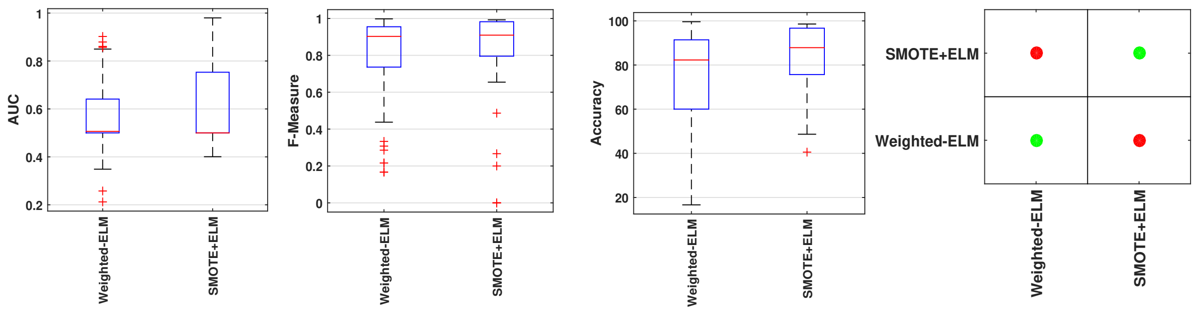

Fig. 8: Box-and-whisker plot: AUC Value of ELM + SMOTE and Weighted ELM

the pictorial representation of descriptive statistics containing Minimum, Maximum, Interquartile range, Degree of dispersion, and Outliers the AUC, F-Measure, and Accuracy for weighted ELM and ELM + SMOTE. An analysis of the Figure 8 indicates the performance parameter value of ELM + SMOTE is better than the one in which we use weighted ELM. So, the aging prediction model developed using ELM + SMOTE obtains better performance as compared to weighted ELM.

Comparison: Null Hypothesis: weighted ELM and ELM + SMOTE: Hypothesis Statistical Significance Testing: In this paper, we have also applied Wilcoxon signed-rank test with a Bonferroni correction for statistical hypothesis testing i.e., NULL Hypothesis: "There is no significant difference in the predictive ability of models trained using different machine learning techniques". The results of the Wilcoxon signed-rank test with a Bonferroni correction of the weighted ELM and ELM + SMOTE are shown in the last sub-figure Figure8, which consists of green and red dots. The rejected null hypothesis is represented using a red dot and the accepted null hypothesis is represented using the green dot.

The null hypothesis in this experiment is that "there is no statistically significant difference between the model developed using weight ELM and model developed using ELM + SMOTE". The results shown in Figure 8 depicts that the cell contains a red dot for weighted ELM and ELM + SMOTE. Based on this result, it may be observed that the aging prediction model developed using weighted ELM and ELM + SMOTE is significantly different.

V. Discussion of results

The proposed study conducted extensive experimentation with the application of the five feature ranking techniques to extract the relevant metrics against 7 projects, to analyse the impact on the accuracy and predictability of SA related Bugs Prediction models when SMOTE (data sampling) + ELM is used in comparison to WELM. Additionally, we examined the performance of sine, sigmoid, radial basis, and triangular basis function kernels in the case of WELM and linear, polynomial, radial basis, and sigmoid basis function kernels in the case of ELM + SMOTE to further investigate the finding with the use of different kernels. The outcome of the empirical experimentation reveals that in the majority of cases, the performance parameter values of feature ranking techniques with ELM + SMOTE were found to be equivalent to or outperformed than with WELM. The employment of OneR feature ranking techniques typically yields results that are competitive. The ELM + SMOTE with the application of Relief feature ranking has the highest F-measure, at 0.93, of all the combinations.

The experimental analysis of the effectiveness of various kernels manifests that the AUC values of models using sine and sigmoid kernels outperform other applied kernels in the case of weighted ELM, whereas for ELM + SMOTE, the sigmoid kernel underperforms other utilised kernels. The study established the improved performance post-implication of kernel techniques with ELM + SMOTE in comparison to weighted ELM. The ELM + SMOTE with linear kernel secures the highest performance among other developed models with 0.75 AUC value, 92.86% Accuracy, and 0.96 F-measure. This study noticed improved AUC, F-measure, and Accuracy values in the descriptive statics based on WELM and ELM + SMOTE results. The Accuracy value of ELM + SMOTE is 87.86, whereas 82.9 for WELM is 82.29, indicating an increase in accuracy of 5.57%.

VI. Threats to Validity

We have also expressed threats to the validity of the proposed work.

i. *Internal Validity:* In relation to internal validity the SA bugs datasets are utilized, to validate the proposed models, which were sourced from the tera-promise data repository. It is important to note that we cannot assert with absolute certainty that the provided data is 100 per cent accurate. However, we have confidence that it was collected consistently. Another factor affecting internal validity is that the sampled data obtained through the use of the SMOTE technique may not precisely reflect the characteristics of actual aging datasets. Both assertions may lead to a potential threat to internal validity because sampled data is used as an input of the trained models and not generalized for testing. However, in the proposed solutions we have been validated with different classification performance parameters such as Accuracy, AUC, and F-Measure, in order to reduce the validation bias.

ii. *Construct Validity:* Many researchers have already developed the SA bugs prediction methods using different sets of source code metrics, as highlighted within the related work section. These works successfully validated the SA bugs that we have also used in this experiment. So, the construct validity threat related to aging or run time failures does not exist.

iii. *External Validity:* The developed models are validated using 07 different datasets that have been designed using procedure language. The finding may vary for projects developed using other programming languages. Hence, a threat to external validity exists in this study. However, the argument setting of developed models helps to reduce the threats to generalizability.

VII. Conclusion

The development of SA prediction model using source code metrics steers the improved software quality and reduces runtime failure. In this paper, empirical experiments have been conducted on seven different applications and proposed to develop early SA bug prediction. The major contributions of this paper are (a) the development of aging prediction models using weighted ELM and ELM + SMOTE with various kernels, (b) the selection of significant right sets of features using different feature ranking techniques, (c) the handling of imbalanced data using SMOTE and weighted elm, and (d) analysis of the performance of the developed model to find the generalized and meaningful conclusion. The experimental assertions of the study are as follows:

- The effectiveness of feature ranking techniques employing ELM + SMOTE exceeds that of feature ranking techniques utilizing weighted ELM.
- The aging prediction model, developed by incorporating diverse metric sets obtained through feature ranking techniques, demonstrates no significant variation.
- The performance of the same kernel functions with ELM + SMOTE is better than kernel functions with weighted ELM.
- The aging prediction models developed using different kernel functions are significantly different.
- The aging prediction model developed using ELM + SMOTE outperforms the weighted ELM approach, and a prediction model based on weighted ELM and ELM + SMOTE exhibit significant dissimilarities.

Thus, we finally conclude that the better value of AUC for the models trained using weighted ELM and ELM + SMOTE confirms that the trained models have the ability to predict SA bugs. These models can be applied to future releases of the SW system's for proactive runtime failure prediction.

VIII. Acknowledgements

This research is funded by TestAIng Solutions Pvt. Ltd.

References

[1] S. R. Chidamber and C. F. Kemerer, “A metrics suite for object oriented design,” *IEEE Transactions on software engineering*, vol. 20, no. 6, pp. 476–493, 1994.

[2] M. K. Thota, F. H. Shajin, P. Rajesh *et al.*, “Survey on software defect prediction techniques,” *International Journal of Applied Science and Engineering*, vol. 17, no. 4, pp. 331–344, 2020.

[3] R. Pietrantuono and S. Russo, “A survey on software aging and rejuvenation in the cloud,” *Software Quality Journal*, vol. 28, no. 1, pp. 7–38, 2020.

[4] R. Matias, B. E. Costa, and A. Macedo, “Monitoring memory-related software aging: An exploratory study,” in *2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops*. IEEE, 2012, pp. 247–252.

[5] S. S. Chouhan, S. S. Rathore, and R. Choudhary, “A study of aging-related bugs prediction in software system,” in *Proceedings of the International Conference on Paradigms of Computing, Communication and Data Sciences*. Springer, 2021, pp. 49–61.

[6] J. Dai, J. Chen, Y. Liu, and H. Hu, “Novel multi-label feature selection via label symmetric uncertainty correlation learning and feature redundancy evaluation,” *Knowledge-Based Systems*, vol. 207, p. 106342, 2020.

[7] A. G. Karegowda, A. Manjunath, and M. Jayaram, “Comparative study of attribute selection using gain ratio and correlation based feature selection,” *International Journal of Information Technology and Knowledge Management*, vol. 2, no. 2, pp. 271–277, 2010.

[8] W. Zong, G.-B. Huang, and Y. Chen, “Weighted extreme learning machine for imbalance learning,” *Neurocomputing*, vol. 101, pp. 229–242, 2013.

[9] K. Li, X. Kong, Z. Lu, L. Wenyin, and J. Yin, “Boosting weighted elm for imbalanced learning,” *Neurocomputing*, vol. 128, pp. 15–21, 2014.

[10] Y. Huang, C. Kintala, N. Kolettis, and N. D. Fulton, “Software rejuvenation: Analysis, module and applications,” in *Twenty-fifth international symposium on fault-tolerant computing. Digest of papers*. IEEE, 1995, pp. 381–390.

[11] J. Alonso, R. Matias, E. Vicente, A. Maria, and K. S. Trivedi, “A comparative experimental study of software rejuvenation overhead,” *Performance Evaluation*, vol. 70, no. 3, pp. 231–250, 2013.

[12] R. Matias and J. Paulo Filho, “An experimental study on software aging and rejuvenation in web servers,” in *30th Annual International Computer Software and Applications Conference (COMPSAC'06)*, vol. 1. IEEE, 2006, pp. 189–196.

[13] J. Araujo, R. Matos, P. Maciel, R. Matias, and I. Beicker, “Experimental evaluation of software aging effects on the eucalyptus cloud computing infrastructure,” in *Proceedings of the middleware 2011 industry track workshop*, 2011, pp. 1–7.

[14] J. Zheng, H. Okamura, L. Li, and T. Dohi, “A comprehensive evaluation of software rejuvenation policies for transaction systems with markovian arrivals,” *IEEE Transactions on Reliability*, vol. 66, no. 4, pp. 1157–1177, 2017.

[15] N. Padhy, R. Singh, and S. C. Satapathy, “Enhanced evolutionary computing based artificial intelligence model for web-solutions software reusability estimation,” *Cluster Computing*, vol. 22, no. 4, pp. 9787–9804, 2019.

[16] S. Sharma and S. Kumar, “Analysis of ensemble models for aging related bug prediction in software systems.” in *ICSOFT*, 2018, pp. 290–297.

[17] M. Khanna, M. Aggarwal, and N. Singhal, “Empirical analysis of artificial immune system algorithms for aging related bug prediction,” in *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1. IEEE, 2021, pp. 692–697.

[18] F. Qin, X. Wan, and B. Yin, “An empirical study of factors affecting cross-project aging-related bug prediction with tlap,” *Software Quality Journal*, vol. 28, no. 1, pp. 107–134, 2020.

[19] A. G. Koru and H. Liu, “An investigation of the effect of module size on defect prediction using static measures,” in *Proceedings of the 2005 workshop on Predictor models in software engineering*, 2005, pp. 1–5.

[20] T. Mende, “Replication of defect prediction studies: problems, pitfalls and recommendations,” in *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*, 2010, pp. 1–10.

[21] T. Mende and R. Koschke, “Revisiting the evaluation of defect prediction models,” in *Proceedings of the 5th International Conference on Predictor Models in Software Engineering*, 2009, pp. 1–10.

[22] A. Bovenzi, D. Cotroneo, R. Pietrantuono, and S. Russo, “Workload characterization for software aging analysis,” in *2011 IEEE 22nd International Symposium on Software Reliability Engineering*. IEEE, 2011, pp. 240–249.