Evolutionary Computing Driven Extreme Learning Machine for Objected Oriented Software Aging Prediction

Shahanawaj Ahamad
drshahwj@gmail.com

College of Computer Science and Engineering
University of Hail, Hail, Saudi Arabia

Summary
To fulfill user expectations, the rapid evolution of software techniques and approaches has necessitated reliable and flawless software operations. Aging prediction in the software under operation is becoming a basic and unavoidable requirement for ensuring the systems' availability, reliability, and operations. In this paper, an improved evolutionary computing-driven extreme learning scheme (ECD-ELM) has been suggested for object-oriented software aging prediction. To perform aging prediction, we employed a variety of metrics, including program size, McCube complexity metrics, Halstead metrics, runtime failure event metrics, and some unique aging-related metrics (ARM). In our suggested paradigm, extracting OOP software metrics is done after pre-processing, which includes outlier detection and normalization. This technique improved our proposed system's ability to deal with instances with unbalanced biases and metrics. Further, different dimensional reduction and feature selection algorithms such as principal component analysis (PCA), linear discriminant analysis (LDA), and T-Test analysis have been applied. We have suggested a single hidden layer multi-feed forward neural network (SL-MFNN) based ELM, where an adaptive genetic algorithm (AGA) has been applied to estimate the weight and bias parameters for ELM learning. Unlike the traditional neural networks model, the implementation of GA-based ELM with LDA feature selection has outperformed other aging prediction approaches in terms of prediction accuracy, precision, recall, and F-measure. The results affirm that the implementation of outlier detection, normalization of imbalanced metrics, LDA-based feature selection, and GA-based ELM can be the reliable solution for object-oriented software aging prediction.

Keywords:
Software Aging Prediction; Extreme Learning Machine; Genetic Algorithm; Software Metrics, Object-oriented Software

1. Introduction

Software engineering is concerned with the design, development, and maintenance of software [1, 2]. The software systems usually run for extended periods of time in order to produce trustworthy computations and valuable decisions based on the running programs and users' specifications. In actuality, however, there is a constant buildup of bugs during the software run time because of program design, improper application environment, and usage pattern, which accumulate errors and finally cause software aging (SA). The aging-related bugs (ARBs) or defects [3] are the prime root cause that makes software systems more susceptible to suffer aging caused failure because of their dynamic errors accumulation across an operational period [9]. These error accumulations can increase performance degradation and lead to system crashes. Unplanned computer system outages are more likely to happen as a result of aging-caused failures than hardware-caused failures [4,5,6]. Aging caused software failures have been observed in numerous events and operating conditions where it has caused money loss, or even human lives [7, 8, 9]. Such bugs in software components introduce incorrect results that adversely affect the decision process and normal intended functions. The early detection of software aging-related bugs (ARBs) is necessary to rejuvenate the system and mitigate the potential negative effects.

However, predicting aging is a time-consuming operation that has become more complicated when dealing with large-scale software, online applications, and other software products. In practice, eliminating entire bugs is even impractical that eventually makes a system probable of being aged after some time. As a result, attempts of research have been undertaken to discover age-related bugs in object-oriented software, that allow for rapid aging prediction, and also the prevention of system crashes. Early aging prediction consequently execution of rejuvenation processes can increase the system's availability, durability, and reliability.

This research has suggested an evolutionary computing-driven ELM learning scheme for aging prediction in object-oriented software. This research has considered some software features such as code complexity, program size, events of faults, and memory availability, these features largely impact the probability of aging. The study provides a multilayered optimization technique for SA prediction, in which pre-processing is used to get the software aging-related metrics, which includes outlier identification and normalization. Different feature selection approaches, including PCA, LDA, and T-Test, have been used to pre-processed metrics in order to achieve time-efficient aging prediction. A single hidden layer multi-feed-forward neural

Manuscript received January 5, 2022
Manuscript revised January 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.1.100

network (SL-MFNN) has been designed to give a basic ELM model, with an evolutionary computing approach termed Adaptive Genetic Algorithm (AGA) used to select and optimize ELM weight and bias parameters. The aging prediction was done using multivariate regression, and the effectiveness was evaluated in terms of precision, accuracy, recall, F-Measures, and G-mean, among other things.

2. Related Work

Software aging have been found in various application scenarios, such as web servers [12], communication systems [13], defense systems [14], mail servers [15], cloud infrastructures [16,17], and in the Linux kernel code [18]. SA concept and aging-related failure were discussed in [19]. Later, authors identified ARBs [20, 21], identified in two classes such as the issues that are easily isolated and other are those defects that emerge regularly in specified functional conditions which may be caused due to error accumulation and complexity. Authors in [22] examined the relationship between the static software features like code complexity, resource consumption and size of code, and aging probability. Researchers in [23] applied resource utilization or usage pattern to examine aging probability where they used neural; networks for aging probability estimation. [24] discovered a technique named as Mann-Kendall test for the purpose. A machine learning method was recently used to forecast aging in a web application [25]. A non-parametric statistical paradigm with an AR model was suggested for web server aging prediction [21]. The time series analysis-based SA prediction was suggested in [26, 27] where ANN [26] and SVM [27] classifiers were applied to perform aging prediction. Considering software metrics as a potential tool to assess aging probability, in [28] the inter-relationship between software metrics and aging probability was examined. Machine learning methods to forecast aging using software metrics were applied in [29], some data mining approaches have been examined in [34]. [30] reported on memory leak-based aging prediction. The stress testing and trend detection approaches for aging prediction were employed in [31].

3. Our Contribution

For the importance of software operations with reliability an early aging prediction might be crucial. Any potential interruption and failure of the software system may be avoided by earlier aging identification. Evolutionary computing-driven extreme learning machine (ECD-ELM) for ARB identification and aging prediction has been suggested in this study. Eventually, the research was carried out in four stages:

1. Data collection,
2. Data pre-processing,
3. Dimensional reduction or feature selection, and
4. Learning and aging prediction.

Fig. 1 represents the overall proposed approach for software aging prediction.

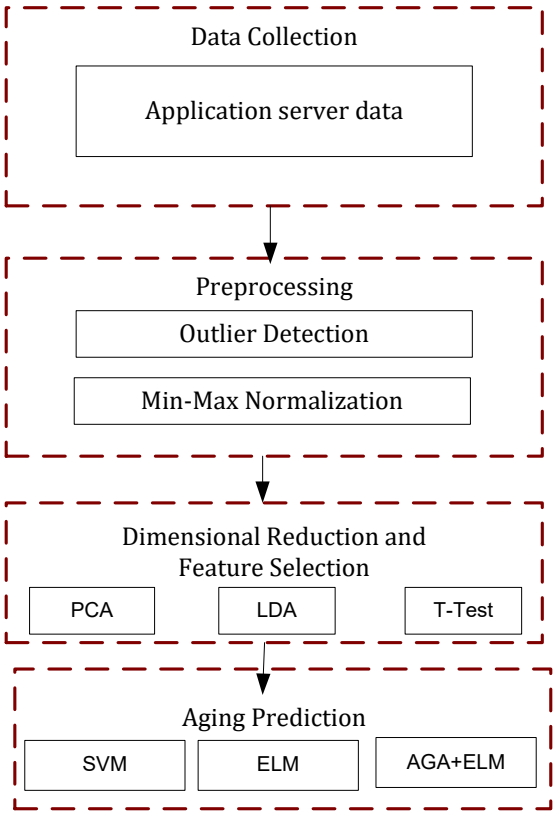

Fig.1 Proposed model for Software aging prediction

A. Data collection

The standard data for "Software Aging and Rejuvenation (SAR)" was initially gathered from two open-source projects, MySQL DBMS and Linux Kernel, and used to accomplish SA prediction in this research. The extracted benchmark data [32] contains various ARBs found in these open-source projects. The prime novelty of our applied datasets is the consideration of a new Aging-Related Metrics (ARM) for aging prediction purposes. In addition, our dataset contains complexity metrics, failure information during run time, etc. The metrics encompass key parameters such as line of code, program declaration, files, statements etc. It encompasses a total of 49 attributes related to the “Program size”, 18 attributes representing “McCabe's cyclomatic complexity metrics”. In addition, we took into account 9 factors related to software's "Halstead" metrics. These metrics refer to operands and operators in the software program. Unlike existing researches, we have introduced six additional metrics called “Aging-related metric (ARM)” for efficient aging analysis. These metrics represent the software failure events during the stress test, memory leak, memory usages in run time.

B. Data Pre-processing

We performed pre-processing on the collected software metrics because the obtained software metrics were often uneven in form. In our study, we first performed outlier detection and elimination, which allows for more efficient data processing. Furthermore, the collected metrics have been normalized to eliminate the problem of unbalanced data. We used the Min-Max normalization technique, which normalizes data in the [-1, 1] range. We wanted to prevent early neural saturation by doing normalization.

C. Dimensional reduction and Feature Selection

There are two major issues that arise in every high-dimensional data-based classification procedure. The first issue is the "curse of dimensionality", whereas the second issue is that many characteristics might have little influence on the aging prediction or ARB classification outcome. These two major concerns have an impact on the whole prediction process in some way. Selecting the appropriate and sufficient features can enable accurate aging prediction. During the software monitoring period, we removed characteristics and data items with fixed values or comparable meanings from our prediction model. We used three different ways to achieve dimensional reduction and feature selection: principal component analysis (PCA), linear discriminant analysis (LDA), and T-Test analysis. The following is a summary of the dimension reduction strategies used:

1. Principal Component Analysis (PCA)

In the applied benchmark SAR data, there can be numerous attributes or parameter elements having similar meaning and constant values. Exclusion of such insignificant or relatively least significant data elements or attributes can make aging prediction swifter and more efficient. In our proposed aging prediction model, the relationship between these parameters and the software aging has been derived as (1):

$$m = f(i_1, i_2, \ldots\ldots, i_n) \tag{1}$$

where m represents the aging probability and $i_1, i_2, \ldots\ldots, i_n$ represent the aging features in software systems. Principal component analysis (PCA) [32] being a multivariate statistical analysis scheme selects multiple principal components to characterize the major changes in the data elements. Here, PCA algorithm is applied for dimensional reduction and feature selection from the input normalized metrics. A brief of PCA implementation is presented as follows:

Consider the data metrics be is$(X = X_1, X_2, \ldots\ldots, X_n)^T$. In our model, two matrixes, the relative matrix (P) and the covariance matrix (C) have been obtained from the sample data. Retrieving these matrixes, the eigenvalues $\lambda_1, \lambda_2, \ldots\ldots, \lambda_n$ and associated eigenvectors have been obtained. With the obtained eigenvectors, we have obtained a new factor called influencing factor (IF) for individual attribute (Program size, line of code, Halstead metrics, McCube cyclomatic complexity, and ARBs). Here, IF indicates the amount to which these characteristics impact the likelihood of software aging (SA). To select principal components, two factors named variance contribution rate (VCR) and cumulative contribution rate (CCR) have been obtained. Mathematically, these factors are given as:

$$VCR_k = \frac{\lambda_k}{(\sum_{i=1}^{n} \lambda_i)} \quad k = (1,2, \ldots, n) \tag{2}$$

Similarly, CCR is estimated as follows:

$$CCR_m = \frac{\sum_{j=1}^{m} \lambda_i}{(\sum_{i=1}^{n} \lambda_i)} \tag{3}$$

where $m = 1,2,3 \ldots, n$.

Higher, VCR means stronger ability of the first principal component (PC) to abstract the information of $x_1, x_2, \ldots\ldots, x_n$. If the CCR of the initial m components is greater than 85%, the initial m components are deemed chosen features for future aging prediction and provide input to the ELM model.

2. Linear Discriminant Analysis (LDA)

In general, PCA uses the most expressive features (MEF) function to pick features; however, MEFs cannot always be the most discriminating features (MDF) function. Furthermore, with the PCA-based technique, a unique PC is created for each class. On the contrary, the linear discriminant analysis (LDA) method provides automated feature selection. To adopt the LDA technique for feature selection, PCA was initially performed, in which all data, regardless of parameter (attributes) category, was projected onto a single PC. Our model yielded two matrices: intra-class scatter matrix I_{nm} and inter-class scatter matrixI_{tB}. Mathematically,

$$I_{nm} = \sum_{i=1}^{p} \sum_{j=1}^{M_i} (y_j - \mu_i)(y_j - \mu_i)^T \tag{4}$$

$$I_{tB} = \sum_{i=1}^{p} (\mu_i - \mu)(\mu_i - \mu)^T \tag{5}$$

where p represents the total number of parameters (attributes or classes) under study, μ_i states the average vector of a class i, and M_i signifies the total number of samples within a class i. Here, the mean μ of all the averaging or mean vectors is obtained as:

$$\mu = \frac{1}{P} \sum_{i=1}^{P} \mu_i \tag{6}$$

Here, LDA emphasizes on maximizing the inter-class scatter while minimizing the intra-class scatter by increasing a ratio$(\det|I_{tB}|/\det|I_{nm}|)$. A prime significance of this ratio is that in case of non-singular I_{tB}matrix, it can be increased when the column vectors of the projection matrix can be the eigenvectors of ${I_{nm}}^{-1} I_{tB}$. In this case, the projection matrix W with $P - 1$ dimension allocates the data onto a new space, commonly referred to as fisher vector, which is then used to do aging prediction.

3. T-Test based feature selection

In addition to the PCA and LDA based feature selection, we have applied *T-Test Analysis* for dimensional reduction and feature selection. T-Test analysis, in contrast to traditional methodologies, seeks to minimize the data dimension by identifying a limited collection of critical features that may enable excellent classification and prediction. To do this, we used a basic criterion on each feature, with the premise that there is no interaction between the features. In our method, we used a two-class problem, with the null hypothesis (H_0) stating that the means of two populations are identical. i.e.; there exist no significant differences between their means and hence signifies the fact that both the features are almost similar. Since, the similar features do not influence the prediction results significantly and therefore such features can be discarded. On contrary, the features having significant differences can affect prediction and therefore are accepted for further processes. In our work, we have accepted and considered the hypothesis H_1. We used the T-Test on each feature, followed by an analysis of the resulting P value (or the absolute values of T-statistics) for each feature as a metric of how successful it is in distinguishing the groups. Using the T-test for all features, the important features were identified, which were then used to conduct GA-based ELM for ARB classification and software aging prediction. In our investigation, we used software aging as the dependent variable, while other features like ARB, Halstead metrics, McCube metrics, ARM, and so on were used as independent variables. Mathematically,

$$Software\ Aging = F(ARB, ARM, Halstead, McCube) \quad (7)$$

D. Software Aging Prediction by Artificial Intelligence Based Learning

A multivariate regression-based analysis was done after conducting feature selection and choosing the data for aging prediction. Understanding the fundamental drawbacks of standard neural network methods, such as local minima and convergence issues, an AGA-based ELM for aging prediction has been created in this research. The suggested ELM-based aging prediction method is described in the following subsections.

1. Extreme Learning Machines

The number of hidden nodes increases as the neural network's input variables and input layers rise. Growing the number of hidden nodes necessitates additional weight parameter estimates, lengthening the process of learning. To address this issue, Huang et al. [33] introduced a new learning scheme called Extreme learning Machine (ELM) that with single-layered multi feed forward neural network (SL-MFNN) facilitates better generalization performance. An enhanced version is used by [10] for software reusability has also motivates to use here also. ELM can provide high-rate learning by selecting hidden nodes randomly and estimating respective weights. Considering this robustness, in our prediction model we have applied ELM algorithm as a classifier. To perform SA prediction, ELM takes selected features as the inputs and performs multivariate regression to predict aging. Fig. 1 presents the schematic flow of the ELM based aging prediction [10][33][35]. Selecting the software metrics or the features, we have fed it as input to the single hidden layer (with L hidden layers) multi-feed forward neural network (SL-MFNN) with X input.

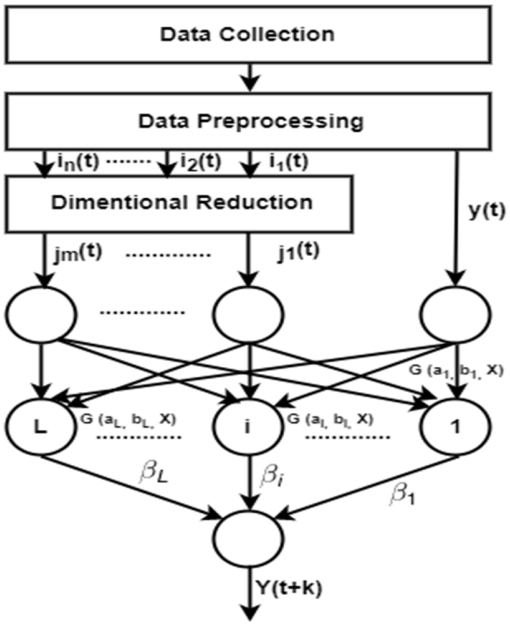

Fig. 2. ECD-ELM for software aging prediction

As depicted in Fig. 2 for evolutionary computing driven ELM (ECD-EML) the input $X = (j_1, j_2, \ldots\ldots, j_m, y)$ denotes a vector with $m+1$ features, $G(a_i, b_i, X)$ indicates the output of the i^{th} hidden neuron where b_i states the bias component of the i^{th} hidden neuron, and $a_i = (a_{i1}, a_{i2}, \ldots\ldots, a_{im}, a_{iy})$ refers the weight vector. Similarly, $a_{is}(s = 1,2, \ldots\ldots, m, y)$ denotes the connection weight between the s^{th} input neuron and the i^{th} hidden neuron. We used AGA, a very resilient and efficient evolutionary computing approach, instead of traditional analytical model-based weight and bias estimation. With the above-mentioned ELM configuration (Fig. 2), the output of the proposed SL-MFNN can be obtained as

$$y(t+k) = f(X) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, X)$$

where, $\beta_i = (\beta_{i1}, \ldots\ldots, \beta_{in}, \beta_{iy})'$ denotes the weight vector connecting hidden layer and output layer. Similarly, β_{ik} represents the connection weight between i^{th} hidden neuron and the k^{th} output neuron. In case of additive hidden neurons, $G(a_i, b_i, X)$can also be represented as follows:

$$G(a_i, b_i, X) = g(a_i'X + b_i) \quad (8)$$

where g: $R \rightarrow R$ represents the activation function.
Unlike conventional analytical schemes for weight estimation, where random weight vectors a_i and bias element b_i are applied to model ELM [33][35], we have applied GA optimization for optimal weight and bias parameter estimation. The SL-MFNN with L hidden neurons can approximate the N samples with zero error only when with specific β_i, we get Y_j as,

$$Y_j = \sum_{i=1}^{L} \beta_i G(a_i, b_i, X_j), j = 1,2, \dots \dots, N \quad (9)$$

In generally, the standard ELM scheme uses an analytical technique to update its weight factor; nevertheless, calculating weights, especially for a large number of input nodes, is extremely difficult. As a result, in this study, we used ECD GA-based optimization to allow optimal weights and bias factor estimates in ELM. ECD GA based ELM parameter optimization is discussed in following sub-sections.

2. *Evolutionary computing driven ELM Learning Weight Estimation*

The selection of weight parameters plays significant role in enabling efficient learning and prediction results. In traditional ELM, particularly with analytical approach-based learning scheme the parameters such as weights and bias components are selected randomly to perform NN learning. However, the random selection of these parameters can't be considered as an optimal solution for efficient learning. Hence, the selection of the optimal learning parameters can make prediction more accurate and sufficient for decision process. With this motivation, in our aging prediction model, we have applied a novel variant of GA algorithm, called Adaptive GA (A-GA). In this case, A-GA was used to determine the best weights and bias parameters for an SL-MFNN-based ELM learning model.
The Genetic Algorithm is a multi-objective approach that operates on the basis of human genetic behavior and the preservation of the fittest concept. It is based on the human evolution principle, which employs the evolutionary notions of natural processes to discover the near-optimal solution with the least amount of computing effort and complexity. The concept that selection of a better population or chromosome for reproduction can enable better next generation with better features and survivability. In function, GA algorithm at first generates random population, where each population represents a solution. GA assesses the fitness value of each chromosome while generating the population, with the greater fitness value of the chromosome indicating a higher selection likelihood. In our proposed model, root mean square error (RMSE) (eq. (17)) has been utilized to assess the fitness value of each chromosome, with the goal of reducing RMSE repeatedly. GA introduces operators based on the collected fitness values such as crossover probability (P_c) and mutation probability (P_m) to find the best solution. Using these operations, the population creation and selection process is repeated until the stopping requirement is reached. Unlike classic GA-based optimization, an adaptive GA (A-GA) method has been devised in this study, which constantly adjusts the genetic activator to ease the problem of local minima and convergence.

Consider the predefined input nodes (or neurons), hidden nodes, output nodes SL-MFNN model be$i$, h and$o$. We have selected 32 features as the input to the ELM model for ARBs classification and bug prediction, after completing the dimensions reduction and attributes features selection. Here, in our ELM model there are 32 input neurons or nodes, one output node and n hidden nodes, where n can be in n to 2^n. In our ELM model, the value of n is 47, and total weights (N) to be estimated for SL-MFNN based EML model would be:

$$N = (i + O) * h = 1551 \quad (10)$$

The estimation of bias parameters is also an intricate task. In traditional ELM models, these parameters (weights and biasing factors are found, which may give inaccurate results for decision process. Therefore, for handling this situation, our proposed A-GA algorithm assists ELM model to select optimal weight and bias parameters for ELM learning and ARBs prediction or aging detection. A brief of A-GA implementation for ELM learning parameter calculation is given as follows:
Let gene length is l, then chromosome length L_{Ch} will be

$$L_{Ch} = N * l = (i + O) * h * l \quad (11)$$

In our model, we have estimated weights (a_i) using following conditions.

$$a_i = \begin{cases} if\ 0 \leq x_{il+1} < 5 \\ -\dfrac{x_{il+2*}10^{l-2} + x_{il+3*}10^{l-3} + \cdots + x_{(i+1)l}}{10^{l-2}} \\ if\ 5 <= x_{il+l} <= 9 \\ +\dfrac{x_{il+2*}10^{l-2} + x_{il+3*}10^{l-3} + \cdots + x_{(i+1)l}}{10^{l-2}} \end{cases} \quad (12)$$

In our proposed ECD-ELM scheme, the genetic parameters (P_c andP_m) are updated dynamically (13).

$$(P_c)_{k+1} = (P_c)_k - \frac{K_1 * n}{7}$$
$$(P_m)_{k+1} = (P_m)_k - \frac{K_2 * n}{7} \quad (13)$$

where $(P_c)_{k+1}$ and $(P_m)_{k+1}$ signifies the crossover and mutation probabilities, respectively. The variables $(P_c)_k$ and $(P_m)_k$ represent the current crossover and mutation probability, while K_1 and K_2 are the positive constant ($K_1 = 0.01$ and $K_2 = 0.001$) and n is number of weights having similar fitness value. In ECD-EML, A-GA continues till 95% of chromosomes are having similar fitness value. As

discussed above, in our model, inverse RMSE as the unbiassed function, where we are purposeful to reduce RMSE iteratively to achieve optimal solution. Now, rewriting equation (9), we get

$$Y = \beta H \tag{14}$$

where

$$Y = \begin{bmatrix} Y_1^T \\ \vdots \\ Y_N^T \end{bmatrix}_{N\times M}, \beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L\times M}$$

and $$H = \begin{bmatrix} G(a_i, b_i, X_1) & \dots & G(a_L, b_L, X_1) \\ \vdots & \dots & \vdots \\ G(a_i, b_i, X_N) & \dots & G(a_L, b_L, X_N) \end{bmatrix}_{N\times L}$$

Once retrieving the learning parameters (i.e., weights and biasing factors) using A-GA, the ELM model with SL-MFNN configuration has been executed for learning. To perform learning we have applied an additional factor called minimum norm least-squares solution (β^*) where we intend to reduceβ^*. Mathematically, β^* is presented as

$$\beta^* = H^+ Y \tag{15}$$

where H^+ represents the inverse of matrix *H*. The overall ELM training process for ARB detection and aging prediction is presented as follows:

Phase-1 *Assign L as hidden layer's number,*
Phase-2 *Assign Activation function* $g()$
Phase-3 *for i =1, 2... L,*
Apply A-GA Based bias (b_i) *and weight* (a_i) *vector estimation*
Phase-4 *Put the estimated vectors to model ELM and calculate the hidden layer (H and*H^+*) output,*
Phase-5 *Estimate the vector* $\beta*$ *as weight output using (15).*
Phase-6 *Estimate output for each sample.*
For any input sample, the output y^* has been estimated as:

$$y^* = \sum_{i=1}^{L} \beta_i^* \, g(a_i x + b_i) \tag{16}$$

Phase-7 Estimate Root means square error (RMSE) as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \left(y(i) - \hat{y}(i)\right)^2}{N}} \tag{17}$$

In eq. (17), $y(i)$ denotes the targeted output, while $\hat{y}(i)$ denotes the outputs. RMSE indicator has been used for aging prediction in this research. In contrast to the ELM and AGA-ELM, we used a linear kernel function-based support vector machine (SVM) in our study model for ARB categorization and aging prediction. Thus, performing multivariate regression between ARBs and aging causing factors, we have obtained confusion metrics, that is further employed to examine aging prediction efficiency.

4. Results and Discussion

Evolutionary computing driven software aging prediction model was developed in this research work. The hypothesis that the continuous accumulation of the bugs either caused by design complexity, memory leak, usages patterns, etc results into software aging, has been considered for study. To examine competence of the AGA-ELM based aging prediction model, at first SAR benchmark data has been retrieved from two open-source project, Linux Kernel and MySQL DBMS. Retrieving software metrics from these open access data, pre-processing is done that reduces data imbalance issue. The implementation of dimensional reduction and feature selection schemes such as PCA, LDA and T-Test analysis has also been done, which has been followed by proposed AGA-ELM based aging prediction. To test the performance AGA-ELM based aging prediction model, two other classification models including traditional ELM and SVM have been applied. The overall software aging prediction model is established using MATLAB R2015a software.

The confusion metrics were created to evaluate the effectiveness of the suggested aging prediction model. The four key variable true positive (TP), false positive (FP), false negative (FN) and true negative (TN) are found and then used to estimate key performance parameter like software aging prediction accuracy, precision, F-Measure, recall and G-Mean etc.

Table 1 Results for MySQL-DBMS SAR data set

Data	Feature Selection	Aging Prediction Technique	Accuracy	Precision	F-Measure	Recall
MySQL DBMS	PCA	SVM	0.829	0.826	0.997	0.839
		ELM	0.849	0.850	0.836	0.823
		AGA-ELM	0.911	0.926	**0.961**	**1.000**
	LDA	SVM	0.794	0.897	0.799	0.721
		ELM	0.867	0.888	0.827	0.775
		AGA-ELM	**0.944**	**0.923**	0.876	0.843
	T-Test	SVM	0.860	0.852	0.865	0.879
		ELM	0.879	0.861	0.834	0.809
		AGA-ELM	0.930	0.913	0.891	0.871

Exploring results in Table 1, we find that with MySQL DBMS SAR data, AGA-ELM outperforms traditional ELM and SVM based aging prediction approaches. Results depict that AGA-ELM gives 94.4% prediction accuracy with LDA feature selection, and is followed by 93.0% accuracy by the same classifier with T-Test analysis based selected features. In addition, AGA-ELM shows 92.3% aging prediction precision with LDA features; however, with PCA it has exhibited slightly elevated precision (92.6%). The proposed AGA-ELM has outperformed other prediction schemes in

terms of F-Measure (96.1%). Similarly, with Linux-Kernel SAR data (Table-2), AGA-ELM has exhibited better aging prediction accuracy (91.3%) with LDA features, precision (92.6%) with T-Test based features and the recall of 96.6% with T-Test features. The relatively better precision and recall with T-Test features has reflected F-measure of 93.0%, which is slightly lower than LDA features (93.6%). Thus, observing overall research outcomes, we find the suggested evolutional computing driven AGA-ELM can be a useful classifier for aging prediction, while LDA can be selected as a feature selection measure for accurate aging prediction based on software metrics. However, T-Test features may be suggested because of its simple and easy to implement feature selection approach.

Table 2 Results for Linux-Kernel SAR data set

Data	Feature Selection	Aging Prediction Technique	Accuracy	Precision	F-Measure	Recall
Linux Kernel	PCA	SVM	0.813	0.867	0.858	0.850
		ELM	0.837	0.881	0.850	0.821
		AGA-ELM	0.899	0.901	0.895	0.889
	LDA	SVM	0.824	0.814	0.797	0.782
		ELM	0.887	0.892	0.850	0.812
		AGA-ELM	**0.913**	0.923	**0.936**	0.941
	T-Test	SVM	0.843	0.810	0.825	0.840
		ELM	0.851	0.821	0.867	0.918
		AGA-ELM	0.889	**0.926**	0.930	**0.966**

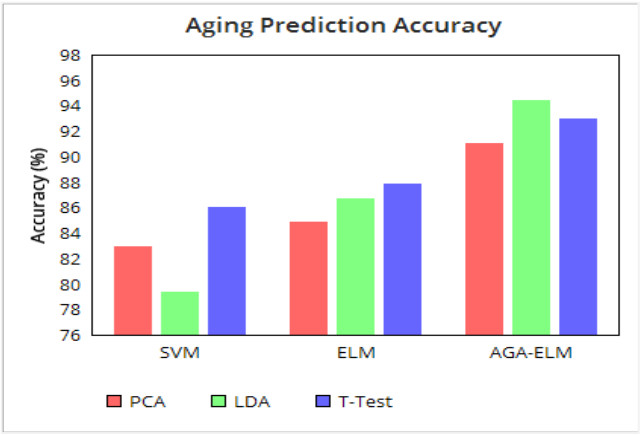

Fig. 3 Software Aging Prediction (SAP) accuracy for MySQL-DBMS Project SAR data

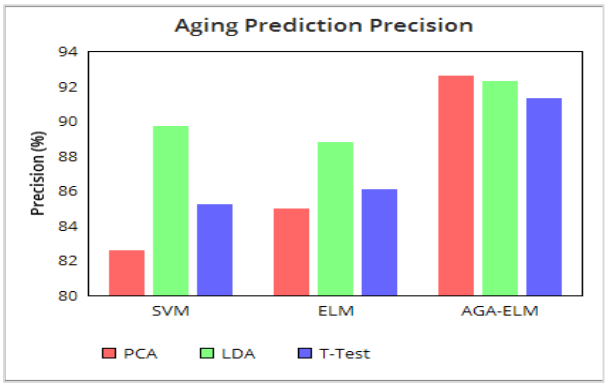

Fig. 4 SAP Precision for MySQL-DBMS Project SAR data

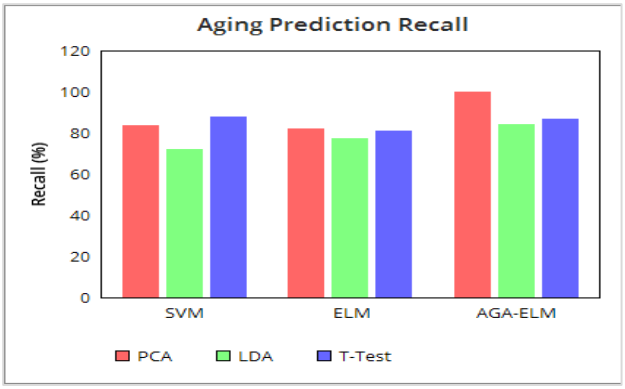

Fig. 5 SAP Recall for MySQL-DBMS Project SAR data

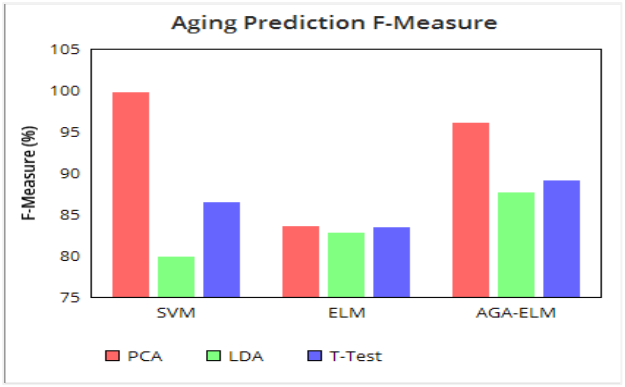

Fig. 6 SAP F-Measure for MySQL-DBMS Project SAR data

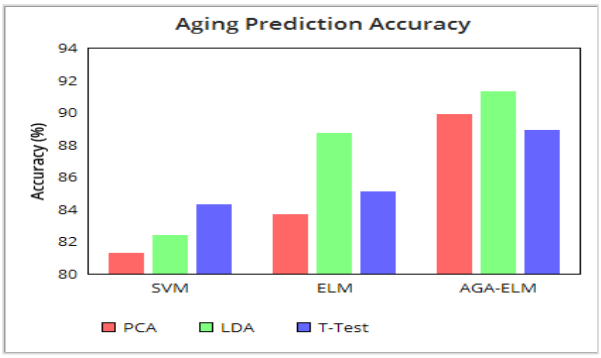

Fig. 7 SAP Accuracy for Linux-Kernel Project SAR data

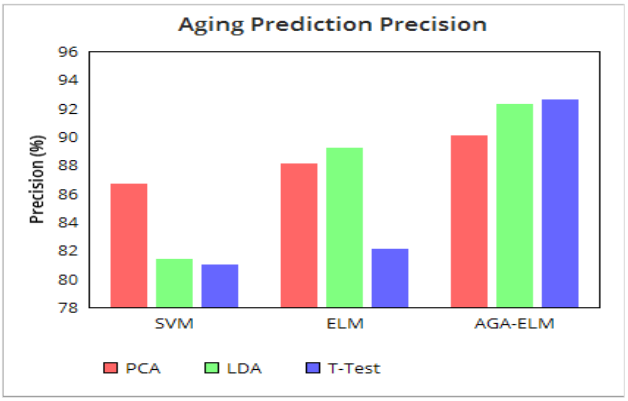

Fig. 8 SAP Precision for Linux-Kernel Project SAR data

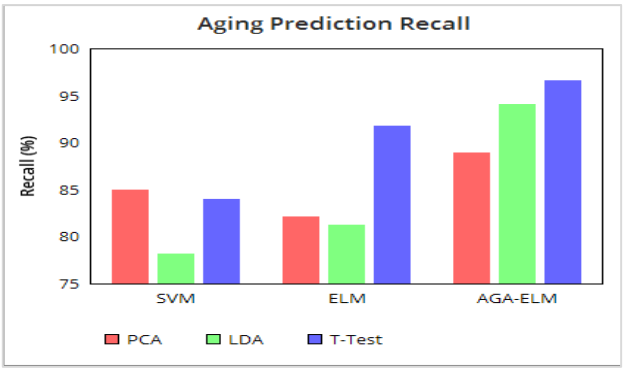

Fig. 9 SAP Recall for Linux-Kernel Project SAR data

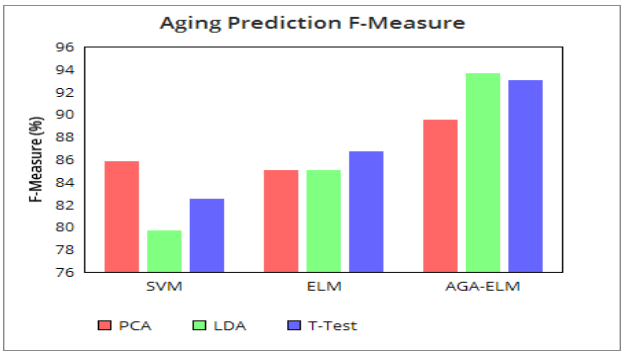

Fig. 10 SAP F-Measure for Linux-Kernel Project SAR data

5. Conclusion

The rapid development of software technologies and services for common decision-making procedures for scientific and general utility situations has become an intrinsic aspect of human life and existence. However, the influence of increasing development and design complexity, operational circumstances, and usage patterns on software aging cannot be overlooked. Software aging results into degraded performance and unexpected failure that eventually results into huge losses regarding service quality, reliability, cost factors or even loss of life. To avoid these, the earlier prediction of software aging is quite necessary. In this paper, evolutionary computing driven software aging prediction scheme was suggested, where to enable optimal prediction the multilevel optimization measures were applied. Here, the deployment of outlier removal and data normalization has facilitated the optimal balanced metrics data for aging prediction in object-oriented software. Considering realistic applications where there can be huge data elements or features, to strengthen performance the use of dimensional reduction and feature selection methods such as PCA, LDA, and T-Test analysis are done. Results obtained with standard Software Aging and Rejuvenation (SAR) datasets, affirm that LDA based feature selection enables better prediction. T-Test implementation, on the other hand, can be a useful measure for enabling rapid and precise aging prediction. According to the deployment of classifiers such as SVM (with linear kernel function), standard ELM, and AGA-ELM, AGA-ELM outperforms other classifiers. This is because the learning process has been enhanced, which was reinforced by applying the appropriate choices for weight and bias parameter. The combinatorial performance confirms that LDA, which frequently provides superior features for subsequent classification when work with AGA-ELM, outperforms other machine and supervised learning schemes, including generic ELM, SVM, and even data mining-based aging prediction methods.

References

[1] S. L. Pfleeger and J. M. Atlee, Software Engineering: Theory and Practice, Prentice Hall; 4 edition, (2009)
[2] Ian Sommerville, Software Engineering, Pearson; 9e (2010)
[3] A. Bovenzi, D. Cotroneo, R. Pietrantuono, S. Russo, Workload Characterization for Software Aging Analysis, in: Proc. IEEE Intl. Symp. On Software Reliability Engineering, pp. 240-249 (2011)
[4] Y. Huang, C. Kintala, N. Kolettis, N. Fulton, Software Rejuvenation: Analysis, Module and Applications, in: Proc. Intl. Symp. on Fault-Tolerant Computing, pp. 381-390, (1995)
[5] M. Balakrishnan, A. Puliaffito, K. Trivedi, I. Viniotisz, Buffer Losses vs. Deadline Violations for ABR Traffic in an ATM Switch: A Computational Approach, Telecommunication Systems 7 (1), 105-123, (1997)

[6] E. Marshall, Fatal Error: How Patriot Overlooked a Scud, Science 255 (5050), 1347, (1992)
[7] M. Grottke, L. Li, K. Vaidyanathan, K. S. Trivedi, Analysis of Software Aging in a Web Server, IEEE Trans. on Reliability 55 (3) , 411-420, (2006)
[8] M. Grottke, R. Matias, K. Trivedi, The Fundamentals of Software Aging, in: Proc. 1st IEEE Intl. Workshop on Software Aging and Rejuvenation, pp. 1-6, (2008)
[9] Ahamad S., Study of software aging issues and prevention solutions. International Journal of Computer Science and Information Security, Aug 1;14(8):307-313, (2016)
[10] Padhy, N., Singh, R. P., & Satapathy, S. C. Enhanced evolutionary computing based artificial intelligence model for web-solutions software reusability estimation. Cluster Computing, 22(4), 9787-9804, (2019).
[11] Kaur, H., Ahamad, S., & Verma, G. N., Elements of Legacy Program Complexity. International Journal of Research in Engineering and Technology, 4(3), 501-505, (2015)
[12] M. Grottke, L. Li, K. Vaidyanathan, K. S. Trivedi, Analysis of Software Aging in a Web Server, IEEE Trans. on Reliability 55 (3), 411-420. (2006)
[13] D. Cotroneo, S. Orlando, R. Pietrantuono, S. Russo, A Measurement based Aging Analysis of the JVM, Software Testing, Verification and Reliability. doi:10.1002/stvr.467.
[14] D. Cotroneo, R. Natella, R. Pietrantuono, S. Russo, Software Aging Analysis of the Linux Operating System, in: Proc. IEEE 21st Intl. Symp. on Software Reliability Engineering, pp. 71-80. (2010)
[15] Grottke, M., K. Trivedi, Software faults, software aging and software rejuvenation, Journal of the Reliability Engineering Association of Japan 27 (7), 425-438. (2005)
[16] S. Garg, A. Pulia_to, K. S. Trivedi, Analysis of Software Rejuvenation using Markov Regenerative Stochastic Petri Net, in: Proc. 6th Intl. Symp. on Software Reliability Engineering, pp. 180-187. (1995)
[17] K. J. Cassidy, K. C. Gross, A. Malekpour, Advanced Pattern Recognition for Detection of Complex Software Aging Phenomena in Online Transaction Processing Servers, in: Proc. IEEE/IFIP Intl. Conf. on Dependable Systems and Networks, pp. 478-482. (2002)
[18] K. Vaidyanathan, K. S. Trivedi, A Measurement-Based Model for Estimation of Resource Exhaustion in Operational Software Systems, in: Proc. 10th Intl. Symp. on Software Reliability Engineering, pp.84-93. (1999)
[19] W. Li and S. Henry, "Maintenance metrics for the Object-Oriented paradigm," in Proceedings of First International Software Metrics Symposium, pp. 52–60, (1993)
[20] M. Grottke, K. Trivedi, Fighting Bugs: Remove, Retry, Replicate, and Rejuvenate, IEEE Computer 40 (2), 107-109. (2007)
[21] R. Matias, P. J. Freitas Filho, An Experimental Study on Software Aging and Rejuvenation in Web Servers, in: Proc. 30th Annual Intl. Computer Software and Applications Conf., pp. 189-196, (2006)
[22] Chug, A., Dhall, S., "Software defect prediction using supervised learning algorithm and unsupervised learning algorithm, "Confluence 2013: The Next Generation Information Technology Summit, pp.173-179, 26-27 Sept. (2013).
[23] F. B. E. Abreu, R. Carapuca, "Object-Oriented software engineering: Measuring and controlling the development process," in Proceedings of the 4th International Conference on Software Quality, vol. 186, (1994)
[24] B. K. Kang and J. M. Bieman, "Cohesion and reuse in an Object-Oriented system," in Proceedings of the ACM SIGSOFT Symposium on software reusability, pp. 259–262, Seattle, March (1995)
[25] L. C. Briand, J. Wust, J. W. Daly, D. V. Porter, "Exploring the relationships between design measures and software quality in Object-Oriented systems," The Journal of Systems and Software, vol. 51, pp. 245–273, (2000)
[26] M. Halstead, Elements of Software Science. New York, USA: Elsevier Science, (1977)
[27] B. Henderson-Sellers, Software Metrics. Prentice-Hall, (1996)
[28] T. J. McCabe, "A complexity measure," IEEE Transactions on Software Engineering, vol. 2, pp. 308–320, (1976)
[29] D. P. Tegarden, S. D. Sheetz, D. E. Monarchi, "A software complexity model of Object-Oriented systems," Decision Support Systems, vol. 13, no. 3, pp. 241–262, (1995)
[30] M. Lorenz and J. Kidd, Object-Oriented Software Metrics. NJ, Englewood: Prentice-Hall, (1994)
[31] S. R. Chidamber and C. F. Kemerer, "A metrics suite for Object-Oriented design," IEEE Transactions on Software Engineering on June 1994, vol. 20, pp. 476–493, (1994)
[32] http://openscience.us/repo/software-aging/
[33] Huang GB, Zhu QY, Siew CK. Extreme Learning Machine: Theory and Applications. Neuro computing, 70(1-3): 489-501, (2006)
[34] Amir Ahmad, predicting software aging related bugs from imbalanced datasets by using data mining techniques, IOSR Journal of Computer Engineering (IOSR-JCE), Volume 18, Issue 1, Ver. III, PP 27-35. (2016)
[35] Xiaozhi Du, Huimin Lu, Gang Liu, Software Aging Prediction based on Extreme Learning Machine, TELKOMNIKA, Vol.11, No.11, pp. 6547-6555 (2013)

Dr. Shahanawaj Ahamad is an active educator and researcher in the field of Computer Science and Software Engineering with 17 years of experience. He completed 3 master's qualifications followed by a Ph.D. degree in Computer Science specializing in Software Engineering; contributed to publish 50 research articles and 3 books. He is designated as Asst. Professor and Program Coordinator of Software Engineering in College of Computer Science and Engineering, University of Hail, Saudi Arabia. He has been contributing significantly to various academic and administrative responsibilities, and a member of several scientific and research organizations including fellowship of British Computer Society, UK. His research interest includes software engineering, software aging and program analysis, application of machine learning, IoT and cloud computing.