# JVLC

## Journal of Visual Language and Computing

Volume 2019, Number 2

Journal of Visual Language and Computing

journal homepage: www.ksiresearch.org/jvlc

# An Empirical Analysis on Effectiveness of Source Code Metrics for Models Predicting Aging Related Bug,★★

Chinmay Hota[a,1], Lov Kumar[a,*,2] and Lalita Bhanu Murthy Neti[a,**]

[a]BITS Pilani Hyderabad Campus, India

ARTICLE INFO

*Article History:*
Submitted 4.30.2019
Revised 5.19.2019
Second Revision 7.29.2019
Accepted 10.10.2019

*Keywords*:
Aging
Software Engineering
Source Code Metrics
Machine Learning
Feature selection

ABSTRACT

Reliability of any software plays a very important role to assess the quality of the software systems. Reliability analysis and evaluation during early phases of software development life cycle (SDLC) significantly help developer and analysts in the proper allocation of limited resources during testing and maintenance process. The goal of this work is to develop one model using the measurement of the internal structure of the software system i.e., source code metrics for predicting the reason of failure in software during continuously running for a certain time i.e., software aging. Aging-Related Bugs are related with failure during the execution of the software, that leads to degradation in quality, system crashing, misuse of resources, etc.. In this paper, seven different sets of software metrics, seven model training algorithms, one data sampling technique have been empirically investigated and evaluated for predicting aging-related bugs classes. The trained aging-related bugs prediction models are validated using 5-fold cross-validation techniques. The final observation of this experiment work is assessed over seven open source application software systems. The high-value of performance parameters confirm the predicting capability of data sampling, sets of metrics, and training algorithms to predict aging-related bugs classes.

## 1. Introduction

Today's scenario, the majority of the software corporation has it in mind to apply the Object-Oriented (OO) tools for construction the contemporary software system due to its efficient blueprint characteristic such as reusability (extending the code use again) and reducing vulnerability, which facilitates more rapid product development [3][5]. With the help of these features, we can achieve the OO properties like-Inheritance, data abstraction, polymorphism, etc.. Software development or solutions of business requirement using OO programming expect high-quality such reliability, maintainability, reusability, portability, etc.. Measuring quality of software and understanding their relationship with software metrics can help tester, architect, and developer control and estimate quality of software by analyzing the source code. The experimental study presented in the paper is motivated by the need to correlate and investigate aging-related bugs work and source code metrics. Aging-related bugs are related with failure during the execution of the software, that leads to degradation in quality, system crashing, misuse of resources, etc.. The effort and cost of fixing aging-related bugs increase exponentially if these bugs are not encountered before the deployment of the software [4][10]. The aging-related bugs prediction models can be applied in the early phase of SDLC. Further, their use can improve the quality of the software as well as reduces the maintenance and testing cost. The experimental study presented in this paper is to develop one model for automatic detection of aging-related bugs work. In particular, our motivation is to investigate the application of different feature selection techniques, classification techniques, and data imbalanced techniques for the task of predicting run time failure i.e., aging-related bugs. Thus, the following research questions are addressed in this work:

- ***RQ1: What is the capability of different classification techniques to train the models for predicting aging-related bugs?*** In this research question, the 5-fold cross-validation technique has been considered to val-

★★ Aging Related Bug Prediction.

*Corresponding author
**Principal corresponding author
f20160069@hyderabad.bits-pilani.ac.in (C. Hota);
lovkumar@hyderabad.bits-pilani.ac.in (L. Kumar);
bhanu@hyderabad.bits-pilani.ac.in (L.B.M. Neti)
ORCID(s):

DOI reference number: 10-18293/JVLC2019-N2-022

**Table 1**
Experimental Dataset Description

| ID | Name | Total classes | non-aging classes | % non-aging classes | aging classes | % aging classes |
|---|---|---|---|---|---|---|
| P1 | Linux driver net | 2292 | 2283 | 99.61 | 9 | 0.39 |
| P2 | Linux ext3 | 29 | 24 | 82.76 | 5 | 17.24 |
| P3 | Linux driver scsi | 962 | 958 | 99.58 | 4 | 0.42 |
| P4 | Linux ipv4 | 117 | 115 | 98.29 | 2 | 1.71 |
| P5 | MySQL optimizer | 36 | 33 | 91.67 | 3 | 8.33 |
| P6 | MySQL replication | 32 | 28 | 87.5 | 4 | 12.5 |
| P7 | MySQL innodb | 402 | 370 | 92.04 | 32 | 7.96 |

idate the models developed using seven different classification techniques. The capability and significance of all considered classification techniques are assessed based on the seven different datasets using rank-sum test and AUC values.

- ***RQ2: What is the effect of different sets of metrics for training the models using classification techniques?*** In this research question, seven different sets of features extracted from source code metrics i.e., software metrics are used as an input to train the aging-related bug prediction models. The capability and significance of these sets of features are computed and compared using AUC values on the seven different datasets.
- ***RQ3: What is the capability of data sampling technique to train the models for predicting aging-related bugs?***

  In this research question, aging-related bug prediction models are trained using balance data and imbalance data. The capability and significance of the data sampling techniques are computed and compared using AUC values on the seven different datasets.

## 2. Related Work

Grottke et al. [4] investigated that aging-related bugs caused software failures have been observed in numerous events and operating conditions, where it has caused major losses with respect to hazards, the human being as well as money.

Junjun Zheng et al. [10] proposed six number of rejuvenation rules, which are particularly meet with the time-based constraints. They demonstrated a systematic study to measure those six software rejuvenation rules numerically and illustrated that the planned rejuvenation rules often termed as wait-time rules are better to the other Markovian arrival processes (MAPs). In practice eliminating entire bugs is even impractical that eventually makes a system probable of being aged after some time. Therefore, efforts must have been made to detect aging-related bugs from software systems to enable swift aging-related bugs detection and to avoid future system-failures. Both bug identification and prediction are the challenging tasks of the developer.

Padhy et al. [8] developed a framework called re-usability optimization based aging-related bugs prediction system, which will estimate the re-usability level. Their focus was on re-usability risk management, where they argued that excessive reused system enters into aging-related bugs. Rejuvenation is one of the practical approaches to avoid failures, which causes the software aging-related bugs. Generally, those aging-related bugs are hard to eliminate completely in the software development as well as testing phase. These types of bugs are often known as a memory leak. The process of handling the software aging-related bugs play the key role and proactive management is required to handle such kind issues.

## 3. RESEARCH BACKGROUND

In this section, the details regrading dataset, sampling technique, techniques used for selecting significant uncorrelated predictors, ELM with three kernels have been summarized.

### 3.1. Experimental Dataset

In this experiment, the empirical results are computed over seven large-scale software applications. These software are mainly used for the development of mission-critical, distributed applications, near real-time applications. This software come in the category of large-scale software used in real-world applications, including safety-critical and business contexts. The data has been collected using github Repository[1]. The github Repository website is a very well-known data repository consisting of various research datasets related to software engineering on defects, code analysis, effort, bugs, maintainability, refactoring, and test generation. The primary objective of using these datasets to replicated our experiment finding, so that these experimental findings are used by other researchers for benchmarking and comparison. In this paper, experiments are conducting on a dataset containing information of aging-related bugs. The dataset contains various types of metrics such as program size related metrics, Halstead metrics, McCabe's Cyclomatic complexity, and aging-related metrics [7]. The characteristics of these datasets are given in Table 1. The data has been collected for seven different projects (Four projects: Linux subsystems and three projects: MySQL sub-systems). From Table 1, it can be clearly seen that Linux driver net is the largest project containing 2292 classes and Linux ext3 is the

[1] https://github.com/lov505/Aging-data

**Table 2**
Software Metrics

| Types of Metrics Sets | Metrics List |
|---|---|
| Program size metrics | AltAvgLineCode, AltAvgLineComment, AltCountLineCode, AvgCyclomatic, AvgCyclomaticModified, AvgCyclomaticStrict, AvgEssential, AvgLineComment, CountClassCoupled, CountDeclClassMethod, CountDeclClassVariable, CountDeclFunction, CountDeclInstanceMethod, CountDeclInstanceVariablePrivate, CountDeclMethod, CountDeclMethodPrivate, CountDeclMethodProtected, CountDeclMethodPublic, CountInput, CountLine, CountLineCode, CountLineCodeDecl, CountLineCodeExe, CountLineComment, CountLineInactive, CountLinePreprocessor, CountSemicolon, CountStmt, CountStmtDecl, CountStmtEmpty, AltAvgLineBlank, AltCountLineBlank, AltCountLineComment, AvgLine, AvgLineBlank, AvgLineCode, CountClassBase, CountClassDerived, CountDeclClass, CountDeclInstanceVariable, CountDeclInstanceVariableProtected, CountDeclInstanceVariablePublic, CountDeclMethodAll, CountDeclMethodConst, CountDeclMethodFriend, CountLineBlank, CountOutput, CountPath, CountStmtExe |
| MCC metrics | CyclomaticStrict, Essential, Knots, MaxCyclomatic, MaxCyclomaticModified, MaxEssentialKnots, MaxInheritanceTree, MaxNesting, MinEssentialKnots, SumCyclomatic, SumCyclomaticModified, SumCyclomaticStrict, Cyclomatic, CyclomaticModified, MaxCyclomaticStrict, PercentLackOfCohesion, RatioCommentToCode, SumEssential |
| Halstead metrics | Number of distinct operands, Program length, Difficulty, Number of distinct operators, Effort, Total number of operands, Volume, Program vocabulary, Total number of operators |
| Aging metrics | DerefUse, UniqueDerefSet, DeallocOps, AllocOps, DerefSet, UniqueDerefUse |

smallest project containing 29 classes among all projects. It is also observed from Table 1 that are considered data are highly imbalanced because % of aging-related classes varies from 0.39% to 17.24%.

### 3.2. Software Metrics

The features extracted from source code called software metrics used to measure the size and complexity of source code. These extracted features are considered as an input to train the models for predicting aging-related bugs. In this work, we have used 49 metrics related to program size, 18 metrics related to McCabe's Cyclomatic complexity, 9 metrics related to Halstead metrics, and 6 metrics related to aging-related bugs as shown in Table 2 for measuring size and complexity of source code. The abbreviation and details explanation of these metrics sets are explained with practical applications in [1][6][2] [2].

### 3.3. Feature Selection Techniques

The next step of this work is to apply feature selection techniques on feature extracted from source code metrics for finding the right sets of uncorrelated significant features. In this step, some important features are selected from all considered features and these selected features are considered by training algorithms to train the models for predicting aging-related bugs. In this paper, we have used univariate rank-sum test and logistic regression to remove insignificant predictor for aging-related bugs. Ramsum test is used to test null hypothesis i.e, no statistical significance between the values of a feature of the class having aging-related bugs and the values of a feature of the class having non-aging-related bugs bugs. The considered null hypothesis has been tested at 0.05 p-value i.e., p-value$\leq$0.05: hypothesis rejected means metrics is relevant for aging-related bugs prediction and p-value>0.05: hypothesis accepted means metrics is irrelevant for aging-related bugs prediction. After removing all insignificant predictors, we are using cross-correlation analysis to remove highly correlated features and finally select significant uncorrelated features. The primary objective of the above step is to select independent significant features having a high correlation with aging-related bugs.

### 3.4. Classification Algorithms:

In this research, we are using extreme learning machine (ELM) algorithm with different kernels functions to train the aging-related bugs prediction models by considering features extracted from source code metrics as input. ELM is based on the concept of randomly generated hidden neurons with an objective to reduce the learning time of the models [9]. ELM algorithm overcomes various problems such as improper learning rate and overfitting, a problem due to local minima, backpropagation. Finally, the trained models using ELM for predicting aging-related bugs are compared with widely used classifiers using AUC value and accuracy.

### 3.5. Data Sampling Technique to handle Imbalanced Data:

Models trained using imbalanced data are the common problem in many domains related to data mining and machine learning such as medical diagnosis, defect prediction, fraud detection etc.. A data is called imbalanced if it has not equal distributions of classes. These types of data sets may be affected the actual performance of models. It can be seen from Table 1 that the percentage of aging-related bugs classes present is considered datasets varies from 0.39% to 17.24 %. Since, the percentage of aging-related bugs classes are not equal to percentage of non-aging-related bugs classes, so models developed using these datasets suffered from class imbalance problem. So, in this work, upscale sampling technique has been considered to deal with class imbalanced class problem. The performance of the developed models using upscale data are computed using accuracy and AUC values and compared with original data.

[2]https://github.com/sealuzh/user_quality/wiki/Code-Quality-Metrics

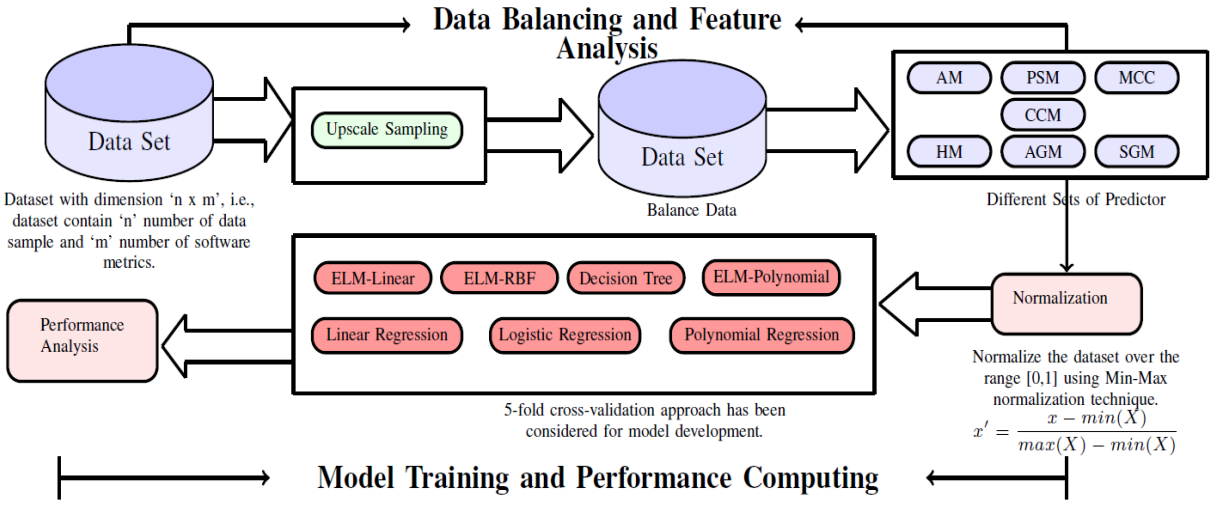


**Figure 1:** Framework of proposed work

### 3.6. Performance Parameter

In order to measure the quality of aging-related bugs prediction models, various performance metrics need to be examined which indicated the effectiveness of the trained models. In this work, we have considered two different performnace metrics such as Accuracy (%) and area under curve value (AUC) for evaluating the quality of aging-related bugs prediction models. A high value of these metrics indicates that the built prediction model is good.

## 4. Research Framework and Experimental Results

This section presents the experimental framework used for developing aging-related bugs prediction models using different sets of source code metrics. Figure 1 shows the empirical framework consisting various steps for development of aging-related bugs prediction model.

From Figure 1, it can be seen that five major steps are used in the proposed framework. The five steps are: (1) Data balancing using upscale sampling technique (2) Categorization of all considered metrics into different sets and also selection of significant and uncorrelated right sets of metrics (3) Data normalization using Min-Max normalization technique (4) Predictive model trained using different classification techniques (5) Significance and capability evaluation using different performance parameters. The above mentioned steps are applied on different packages of two long-living, complex, and large software systems i.e., MySQL and Linux kernel. In this work, eighty two different metrics such as Halstead metrics, McCabe's cyclomatic complexity, size related metrics, and aging-related bugs metrics have been considered as predictors for aging-related bugs prediction models. From Table 1, we can observed that the considered datasets are suffering from class imbalance problem. So, we have considered upscale sampling technique to address the issue of class imbalance problem. The second step consists of Categorization of all considered metrics into different sets i.e., Halstead metrics (HM), McCabe's cyclomatic complexity (MCC), size related metrics (PSM), aging-related bugs metrics (AGM), and selection of right set of significant predictor (SGM) and uncorrelated significant predictor (CCM) using rank-sum test and cross correlation analysis. Then, Min-Max Normalization has applied before training the model to normalize the all selected features in the same range of 0 to 1. Further, seven different classification techniques have been used to train the aging-related bugs prediction model. These models are validated using 5-fold cross-validation approach. Finally, accuracy, and AUC values are computed for each model to evaluate capability and significance of above considered classification techniques, data sampling technique, and different sets of metrics.

### 4.1. Feature Selection Techniques:

In this section, we have summarized the results obtained after rank-sum test on original data and upscale sampling data for different projects. In this study, file having at least one aging-related bugs bugs considered as aging-related bugs classes else non-aging-related bugs classes. Table 3 provides the relevant metrics related program size metrics after rank-sum test for two projects on without sampling data i.e., original data and upscale data. Similarly, the relevant features for other metrics sets are computed using rank-sum test. In Table 3, only 0 and 1 are used to represent relevant features and irrelevant metrics i.e., cell containing 1 denotes the relevant metrics for project using original data or upscale data and cell containing 0 denotes the irrelevant metrics for project using original data or upscale data. It can be observed from Table 3 that AltAvgLineBlank is relevant for all cases except scsi, ext3, and ipv4 projects. Similarly, It can also observed from Table 3 that CountClassBase, CountClassCoupled, CountClassDerived, CountDeclClassMethod, CountDeclClassVariable etc. are irrelevant for aging-related bugs

**Table 3**
Rank-sum Test for Program Size Metrics

| | Original Data | | | | | | | Upscale Sampling | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Linux | | | | MySQL | | | Linux | | | | MySQL | | |
| | net | scsi | ext3 | ipv4 | *innodb* | *optimizer* | *replication* | net | scsi | ext3 | ipv4 | *innodb* | *optimizer* | *replication* |
| AltAvgLineBlank | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| AltAvgLineCode | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| AltAvgLineComment | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| AltCountLineBlank | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| AltCountLineCode | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| AltCountLineComment | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| AvgCyclomatic | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| AvgCyclomaticModified | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| AvgCyclomaticStrict | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| AvgEssential | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| AvgLine | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| AvgLineBlank | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| AvgLineCode | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| AvgLineComment | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| CountClassBase | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CountClassCoupled | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CountClassDerived | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CountDeclClass | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| CountDeclClassMethod | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CountDeclClassVariable | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CountDeclFunction | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| CountDeclInstanceMethod | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CountDeclInstanceVariable | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CountDeclInstanceVariablePrivate | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CountDeclInstanceVariableProtected | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CountDeclInstanceVariablePublic | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CountDeclMethod | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CountDeclMethodAll | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CountDeclMethodConst | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CountDeclMethodFriend | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CountDeclMethodPrivate | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CountDeclMethodProtected | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CountDeclMethodPublic | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CountInput | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CountLine | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| CountLineBlank | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| CountLineCode | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| CountLineCodeDecl | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| CountLineCodeExe | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| CountLineComment | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| CountLineInactive | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| CountLinePreprocessor | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| CountOutput | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CountPath | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| CountSemicolon | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| CountStmt | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| CountStmtDecl | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| CountStmtEmpty | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| CountStmtExe | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

prediction i.e., these metrics are not considered as input to train the aging-related bugs prediction models.

## 4.2. Results and analysis

We have considered seven sets of source code metrics as input to train the model using seven different classification techniques to predict aging-related bugs. Five fold cross validation method has been used to validate the trained models

**Table 4**
Accuracy and AUC values for Linux net and Scsi

| | Without Sampling (net) | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Accuracy | | | | | | | AUC | | | | | | |
| | LR | PR | LOR | DT | ELMLIN | ELMPLY | ELMRBF | LR | PR | LOR | DT | ELMLIN | ELMPLY | ELMRBF |
| Fold1 | 98.69 | 94.32 | 98.25 | 99.78 | 99.78 | 93.45 | 99.78 | 0.49 | 0.47 | 0.49 | 0.50 | 0.50 | 0.47 | 0.50 |
| Fold2 | 98.47 | 95.64 | 98.47 | 99.56 | 99.35 | 76.47 | 99.56 | 0.49 | 0.73 | 0.49 | 0.50 | 0.50 | 0.63 | 0.50 |
| Fold3 | 97.82 | 89.11 | 98.91 | 98.69 | 98.91 | 88.24 | 99.56 | 0.74 | 0.70 | 0.50 | 0.50 | 0.50 | 0.69 | 0.50 |
| Fold4 | 98.69 | 93.23 | 98.47 | 99.56 | 99.56 | 89.52 | 99.56 | 0.99 | 0.97 | 0.74 | 0.50 | 0.50 | 0.45 | 0.50 |
| Fold5 | 97.16 | 92.14 | 97.82 | 99.13 | 99.13 | 96.94 | 99.56 | 0.74 | 0.96 | 0.49 | 0.50 | 0.50 | 0.49 | 0.50 |
| | Upscale Sampling (net) | | | | | | | | | | | | | |
| | LR | PR | LOR | DT | ELMLIN | ELMPLY | ELMRBF | LR | PR | LOR | DT | ELMLIN | ELMPLY | ELMRBF |
| Fold1 | 50.00 | 50.00 | 98.47 | 99.78 | 97.16 | 64.11 | 50.00 | 0.50 | 0.50 | 0.98 | 1.00 | 0.97 | 0.64 | 0.50 |
| Fold2 | 49.95 | 49.95 | 98.47 | 99.56 | 95.84 | 54.87 | 49.95 | 0.50 | 0.50 | 0.98 | 1.00 | 0.96 | 0.55 | 0.50 |
| Fold3 | 50.00 | 50.00 | 99.02 | 99.34 | 96.72 | 67.72 | 50.00 | 0.50 | 0.50 | 0.99 | 0.99 | 0.97 | 0.68 | 0.50 |
| Fold4 | 50.00 | 50.00 | 98.57 | 99.23 | 95.18 | 69.30 | 50.00 | 0.50 | 0.50 | 0.99 | 0.99 | 0.95 | 0.69 | 0.50 |
| Fold5 | 50.05 | 50.05 | 98.69 | 99.67 | 95.51 | 68.02 | 50.05 | 0.50 | 0.50 | 0.99 | 1.00 | 0.96 | 0.68 | 0.50 |
| | Without Sampling (scsi) | | | | | | | | | | | | | |
| | LR | PR | LOR | DT | ELMLIN | ELMPLY | ELMRBF | LR | PR | LOR | DT | ELMLIN | ELMPLY | ELMRBF |
| Fold1 | 95.31 | 76.04 | 97.40 | 100.00 | 99.48 | 82.81 | 100.00 | 0.50 | 0.50 | 0.50 | 0.50 | 0.50 | 0.50 | 0.50 |
| Fold2 | 92.75 | 76.68 | 97.93 | 98.96 | 98.96 | 37.31 | 99.48 | 0.47 | 0.88 | 0.49 | 0.99 | 0.50 | 0.68 | 0.50 |
| Fold3 | 93.75 | 86.98 | 97.40 | 99.48 | 98.44 | 86.46 | 99.48 | 0.47 | 0.44 | 0.49 | 0.50 | 0.49 | 0.93 | 0.50 |
| Fold4 | 95.85 | 79.79 | 98.96 | 99.48 | 99.48 | 95.85 | 99.48 | 0.48 | 0.40 | 0.50 | 0.50 | 0.50 | 0.48 | 0.50 |
| Fold5 | 96.35 | 93.23 | 96.88 | 98.96 | 97.92 | 76.04 | 99.48 | 0.48 | 0.47 | 0.49 | 0.50 | 0.49 | 0.88 | 0.50 |
| | Upscale Sampling (scsi) | | | | | | | | | | | | | |
| | LR | PR | LOR | DT | ELMLIN | ELMPLY | ELMRBF | LR | PR | LOR | DT | ELMLIN | ELMPLY | ELMRBF |
| Fold1 | 50.00 | 50.00 | 94.27 | 99.22 | 93.49 | 56.51 | 50.00 | 0.50 | 0.50 | 0.94 | 0.99 | 0.93 | 0.57 | 0.50 |
| Fold2 | 50.00 | 50.00 | 92.93 | 99.74 | 95.03 | 63.35 | 50.00 | 0.50 | 0.50 | 0.93 | 1.00 | 0.95 | 0.63 | 0.50 |
| Fold3 | 50.00 | 50.00 | 95.83 | 99.48 | 93.49 | 75.26 | 50.00 | 0.50 | 0.50 | 0.96 | 0.99 | 0.93 | 0.75 | 0.50 |
| Fold4 | 49.87 | 49.87 | 93.99 | 98.69 | 93.47 | 53.26 | 49.87 | 0.50 | 0.50 | 0.94 | 0.99 | 0.93 | 0.53 | 0.50 |
| Fold5 | 50.13 | 50.13 | 96.34 | 98.69 | 92.43 | 68.41 | 50.13 | 0.50 | 0.50 | 0.96 | 0.99 | 0.92 | 0.68 | 0.50 |

and performance of these trained models are express in terms of area under curve value (AUC) and accuracy. The value of Accuracy and AUC of trained model using different techniques for Linux net project on each fold shown in Table 4. Similarly, the Accuracy and AUC values for other projects are computed. It can be seen from Table 4 that:

- Aging-related bugs prediction models trained using Upscale sampling data have relatively better AUC values as compare to original data.
- Aging-related bugs prediction models trained using logistic regression and decision tree have relatively better AUC values as compare to other techniques.
- Aging-related bugs prediction models trained using ELM with linear kernel function have relatively better AUC values as compare to other kernel functions.

## 5. Comparison

***RQ1: RQ1: what is the overall predictive capability of model trained using different classification techniques to predict aging-related bugs?*** In this research question, 5-fold cross validation technique has been considered to validate the models developed using seven different classification techniques. The significance and capability of all considered classification techniques are assessed based on the seven different datasets using Boxplots, Descriptive Statistics, and Statistical tests analysis.

*Boxplots and Descriptive Statistics: Classification Techniques* In this paper, boxplot diagram has been considered for visual comparison showing distribution, variability, outliers, degree of dispersion of different classification techniques as depicted in Figure 2. The first, second , and third sub-figures of Figure 2 shows the boxplot diagram of AUC value for classification techniques trained using original data, upscale sampling data, and overall. The median value of AUC value for each classification technique has been presented using red line. The descriptive statistics i.e, **Min**, **25%**, **Mean**, **Median**, **75%**, **Max** for each box are presented in Table 5. It can be seen from Table 5 and Figure 2 that aging-related bugs prediction models trained using decision tree (DT) have relatively better AUC values as compare to other techniques. It can be also seen from Table 5 and Figure 2 that ELM with liner kernel on upscale sampling data and over all provided good results as compared to other polynomial kernel and RBF kernels.

*Comparison of different Classification techniques using Statistical tests:* In this work, we have also considered pairwise rank sum test with Bonferroni correction to compare the performance of developed aging-related bugs prediction models using different classification techniques. We have considered seven sets of metrics, and two different datasets for each seven project. Hence, each classification technique

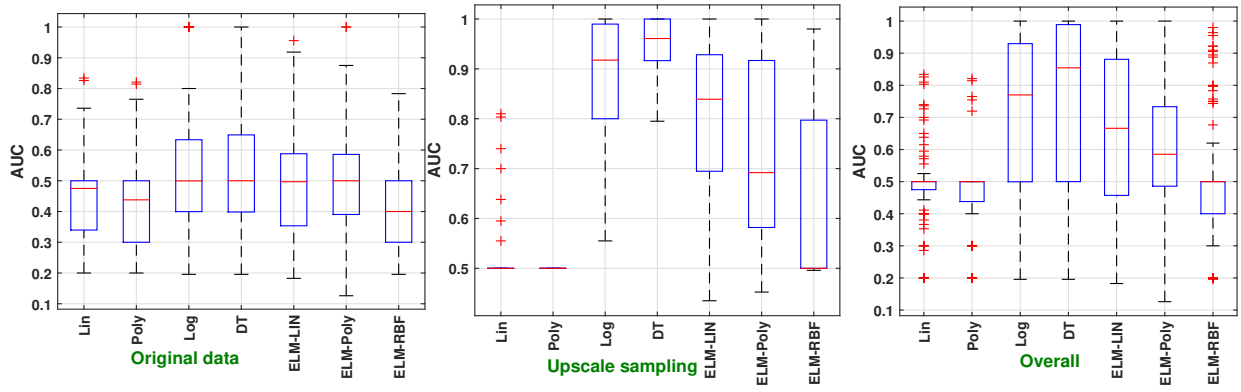


**Figure 2:** Boxplots for AUC for Classification Techniques

has 490 data points or AUC values [7 sets of features * (1 original data + 1 upscale sampling data) * 7 projects * 5 folds]. The null hypothesis of the rank sum test is that *aging-related bugs prediction models developed using different classification techniques are not significantly different*. The rank sum test with Bonferroni correction adjust the threshold p-value at $\frac{0.05}{\text{number of different pairs}}$. Since, we have considered seven different classification techniques, so the number of different pairs is $7c_2 = 7 * 6/2 = 21$. Hence, null hypothesis is accepted only if the p-value is less then $\frac{0.05}{21} = 0.0024$. The results of rank sum test with Bonferroni correction for different classification techniques are summarized in Table 6. We observe from Table 6 that the p-value of 13 out 21 pairs are less than 0.0024 i.e., hypothesis rejected means model trained using different classification techniques are significantly different.

**Table 5**
Statistical Measures on AUC: Classification Techniques

| | Original Data | | | | | |
|---|---|---|---|---|---|---|
| | **Min** | **Max** | **Mean** | **Median** | **25%** | **75%** |
| LR | 0.20 | 0.83 | 0.45 | 0.48 | 0.34 | 0.50 |
| PR | 0.20 | 0.82 | 0.43 | 0.44 | 0.30 | 0.50 |
| LOR | 0.20 | 1.00 | 0.54 | 0.50 | 0.40 | 0.63 |
| DT | 0.20 | 1.00 | 0.55 | 0.50 | 0.40 | 0.65 |
| ELMLIN | 0.18 | 0.96 | 0.49 | 0.50 | 0.35 | 0.59 |
| ELMPLY | 0.13 | 1.00 | 0.50 | 0.50 | 0.39 | 0.59 |
| ELMRBF | 0.20 | 0.78 | 0.41 | 0.40 | 0.30 | 0.50 |
| | **Upscale Sampling** | | | | | |
| | **Min** | **Max** | **Mean** | **Median** | **25%** | **75%** |
| LR | 0.50 | 0.81 | 0.53 | 0.50 | 0.50 | 0.50 |
| PR | 0.50 | 0.50 | 0.50 | 0.50 | 0.50 | 0.50 |
| LOR | 0.56 | 1.00 | 0.88 | 0.92 | 0.80 | 0.99 |
| DT | 0.80 | 1.00 | 0.95 | 0.96 | 0.92 | 1.00 |
| ELMLIN | 0.44 | 1.00 | 0.81 | 0.84 | 0.69 | 0.93 |
| ELMPLY | 0.45 | 1.00 | 0.73 | 0.69 | 0.58 | 0.92 |
| ELMRBF | 0.50 | 0.98 | 0.63 | 0.50 | 0.50 | 0.80 |
| | **Overall** | | | | | |
| | **Min** | **Max** | **Mean** | **Median** | **25%** | **75%** |
| LR | 0.20 | 0.83 | 0.49 | 0.50 | 0.48 | 0.50 |
| PR | 0.20 | 0.82 | 0.47 | 0.50 | 0.44 | 0.50 |
| LOR | 0.20 | 1.00 | 0.71 | 0.77 | 0.50 | 0.93 |
| DT | 0.20 | 1.00 | 0.75 | 0.85 | 0.50 | 0.99 |
| ELMLIN | 0.18 | 1.00 | 0.65 | 0.67 | 0.46 | 0.88 |
| ELMPLY | 0.13 | 1.00 | 0.62 | 0.59 | 0.49 | 0.73 |
| ELMRBF | 0.20 | 0.98 | 0.52 | 0.50 | 0.40 | 0.50 |

**Table 6**
P-value: Classification Techniques

| | **LR** | **PR** | **LOR** | **DT** | **ELMLIN** | **ELMPLY** | **ELMRBF** |
|---|---|---|---|---|---|---|---|
| LR | 1.00 | 0.17 | 0.00 | 0.00 | 0.00 | 0.00 | 0.98 |
| PR | 0.17 | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.36 |
| LOR | 0.00 | 0.00 | 1.00 | 0.23 | 0.08 | 0.01 | 0.00 |
| DT | 0.00 | 0.00 | 0.23 | 1.00 | 0.00 | 0.00 | 0.00 |
| ELMLIN | 0.00 | 0.00 | 0.08 | 0.00 | 1.00 | 0.23 | 0.00 |
| ELMPLY | 0.00 | 0.00 | 0.01 | 0.00 | 0.23 | 1.00 | 0.00 |
| ELMRBF | 0.98 | 0.36 | 0.00 | 0.00 | 0.00 | 0.00 | 1.00 |

***RQ2: what is the overall predictive capability of model developed using different sets of metrics as input to predict aging-related bugs?*** In this paper,seven different sets of metrics are considered as input to developed a model for predicting aging-related bugs. The significance and capability of all considered sets of metrics are assessed based on the seven different datasets using Boxplots, Descriptive Statistics, and Statistical tests analysis.

*Comparison of different sets of metrics using Boxplots and Descriptive Statistics:* In this paper, boxplot diagram has been considered for visual comparison showing distribution, variability, outliers, degree of dispersion of different sets of metrics as depicted in Figure 3. The first, second , and third sub-figures of Figure 3 shows the boxplot diagram of AUC value for classification techniques trained using original data, upscale sampling data, and overall. The median value of AUC value for each classification technique

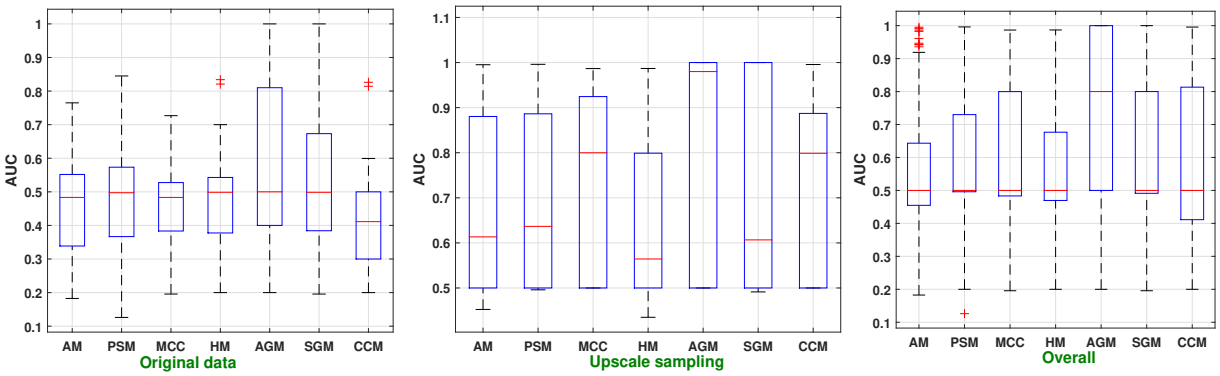


**Figure 3:** Boxplots for AUC for Sampling Technique

has been presented using red line. The descriptive statistics i.e, **Min**, **25%**, **Mean**, **Median**, **75%**, **Max** for each box are presented in Table 7. It can be seen from Table 7 and Figure 3 that aging-related bugs prediction models developed using aging-related bugs (AGM) metrics as input have relatively better AUC values as compare to other sets of metrics. It can be also seen from Table 7 and Figure 3 that significant sets of features (SGM) provided results as compared to other sets of metrics except AGM.

*Comparison of different sets of metrics using Statistical tests:* In this work, we have also considered pairwise rank sum test with Bonferroni correction to compare the performance of developed aging-related bugs prediction models using different sets of metrics. We have considered seven different classification techniques, and two different datasets for each seven project. Hence, each each set of metrics has 490 data points or AUC values [7 classification techniques * (1 original data + 1 upscale sampling data) * 7 projects * 5 folds]. The null hypothesis of the rank sum test is that *aging-related bugs prediction models developed using different sets of metrics are not significantly different*. The rank sum test with Bonferroni correction adjust the threshold p-value at $\frac{0.05}{\text{number of different pairs}}$. Since, we have considered seven sets of metrics, so the number of different pairs is $7c_2 = 7 * 6/2 = 21$. Hence, null hypothesis is accepted only if the p-value is less then $\frac{0.05}{21} = 0.0024$. The results of rank sum test with Bonferroni correction for sets of metrics are summarized in Table 8. We observe from Table 8 that the p-value of the models trained using aging-related bugs metrics are less than 0.0024 i.e., hypothesis rejected means model trained using aging-related bugs metrics and other sets of metrics are significantly different.

***RQ3: what is the overall predictive capability of model developed using balance and imbalance data to predict aging-related bugs?***

In this work, aging-related bugs prediction models are trained using balance data and imbalance data. The significance and capability of the data sampling used for balancing data is assessed based on the seven different datasets using rank-sum test and AUC values. .

**Table 7**
Statistical Measures on AUC: Sets of Metrics.

| | Min | Max | Mean | Median | 25% | 75% |
|---|---|---|---|---|---|---|
| **Original Data** | | | | | | |
| AM | 0.18 | 0.76 | 0.45 | 0.48 | 0.34 | 0.55 |
| PSM | 0.13 | 0.85 | 0.46 | 0.50 | 0.37 | 0.57 |
| MCC | 0.20 | 0.73 | 0.44 | 0.48 | 0.38 | 0.53 |
| HM | 0.20 | 0.83 | 0.46 | 0.50 | 0.38 | 0.54 |
| AGM | 0.20 | 1.00 | 0.61 | 0.50 | 0.40 | 0.81 |
| SGM | 0.20 | 1.00 | 0.52 | 0.50 | 0.38 | 0.67 |
| CCM | 0.20 | 0.83 | 0.42 | 0.41 | 0.30 | 0.50 |
| **Upscale Sampling** | | | | | | |
| | **Min** | **Max** | **Mean** | **Median** | **25%** | **75%** |
| AM | 0.45 | 1.00 | 0.67 | 0.61 | 0.50 | 0.88 |
| PSM | 0.50 | 1.00 | 0.69 | 0.64 | 0.50 | 0.89 |
| MCC | 0.50 | 0.99 | 0.75 | 0.80 | 0.50 | 0.92 |
| HM | 0.44 | 0.99 | 0.64 | 0.56 | 0.50 | 0.80 |
| AGM | 0.50 | 1.00 | 0.83 | 0.98 | 0.50 | 1.00 |
| SGM | 0.49 | 1.00 | 0.70 | 0.61 | 0.50 | 1.00 |
| CCM | 0.50 | 1.00 | 0.74 | 0.80 | 0.50 | 0.89 |
| **Overall** | | | | | | |
| | **Min** | **Max** | **Mean** | **Median** | **25%** | **75%** |
| AM | 0.18 | 1.00 | 0.56 | 0.50 | 0.45 | 0.64 |
| PSM | 0.13 | 1.00 | 0.58 | 0.50 | 0.50 | 0.73 |
| MCC | 0.20 | 0.99 | 0.60 | 0.50 | 0.48 | 0.80 |
| HM | 0.20 | 0.99 | 0.55 | 0.50 | 0.47 | 0.68 |
| AGM | 0.20 | 1.00 | 0.72 | 0.80 | 0.50 | 1.00 |
| SGM | 0.20 | 1.00 | 0.61 | 0.50 | 0.49 | 0.80 |
| CCM | 0.20 | 1.00 | 0.58 | 0.50 | 0.41 | 0.81 |

*Comparison of different sampling techniques using Boxplots and Descriptive Statistics:* In this paper, boxplot diagram has been considered for visual comparison showing distribution, variability, outliers, degree of dispersion of original data and upscale sampling data as depicted in Figure 4. The median value of AUC value for each data set has been presented using red line. The descriptive statistics i.e, **Min**, **25%**, **Mean**, **Median**, **75%**, **Max** for each box are presented

**Table 8**
P-value: Sets of Metrics

| | AM | PSM | MCC | HM | AGM | SGM | CCM |
|---|---|---|---|---|---|---|---|
| AM | 1.00 | 0.67 | 0.41 | 0.87 | 0.00 | 0.17 | 0.73 |
| PSM | 0.67 | 1.00 | 0.65 | 0.52 | 0.00 | 0.33 | 0.96 |
| MCC | 0.41 | 0.65 | 1.00 | 0.26 | 0.00 | 0.51 | 0.59 |
| HM | 0.87 | 0.52 | 0.26 | 1.00 | 0.00 | 0.15 | 0.59 |
| AGM | 0.00 | 0.00 | 0.00 | 0.00 | 1.00 | 0.01 | 0.00 |
| SGM | 0.17 | 0.33 | 0.51 | 0.15 | 0.01 | 1.00 | 0.31 |
| CCM | 0.73 | 0.96 | 0.59 | 0.59 | 0.00 | 0.31 | 1.00 |

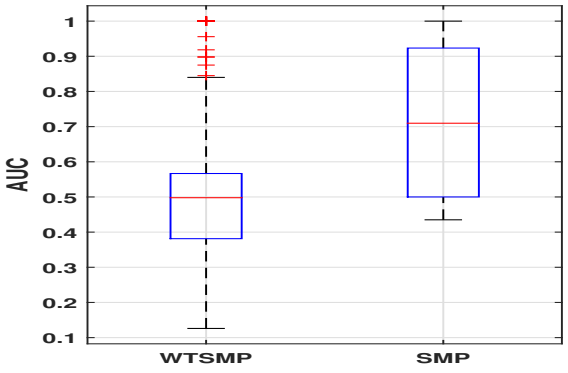


**Figure 4:** Boxplots for AUC

in Table 9. It can be seen from Table 9 and Figure 4 that aging-related bugs prediction models developed using sampling data i.e., balance data have relatively better AUC values as compare to original data. *Comparison of different sampling techniques using Statistical tests:* In this work, we have also considered pairwise rank sum test with Bonferroni correction to compare the performance of developed aging-related bugs prediction models using original data and sampled data. We have considered seven different classification techniques, seven sets of metrics, and seven project. Hence, each data set has 1715 data points or AUC values [7 classification techniques * 7 sets of features * 7 projects * 5 folds]. The null hypothesis of the rank sum test is that *aging-related bugs prediction models developed using original data and sampled data are not significantly different*. The rank sum test with Bonferroni correction adjust the threshold p-value at $\frac{0.05}{\text{number of different pairs}}$. Since, we have considered two datasets, so the number of different pairs is $2c_2 = 1$. Hence, null hypothesis is accepted only if the p-value is less then $\frac{0.05}{1} = 0.05$. The results of rank sum test with Bonferroni correction for sets of metrics are summarized in Table 10.

**Table 9**
Statistical Measures on AUC: Original data and Upscale Data

| | Min | Max | Mean | Median | 25% | 75% |
|---|---|---|---|---|---|---|
| WTSMP | 0.13 | 1.00 | 0.48 | 0.50 | 0.38 | 0.57 |
| SMP | 0.44 | 1.00 | 0.72 | 0.71 | 0.50 | 0.92 |

We observe from Table 10 that the p-value of the models trained using original data and sampled data are significantly different

**Table 10**
P-value: Original data and Upscale Data

| | WTSMP | SMP |
|---|---|---|
| WTSMP | 1.00 | 0.00 |
| SMP | 0.00 | 1.00 |

## 6. Conclusion

The primary goal of this research is to find the effect of internal structure of software on aging-related bugs. In this experiment, we also empirically evaluated, analyzed, and compared the performance of developed aging-related bugs prediction models using different classification techniques, sets of metrics, selected set of metrics, and data sampling technique. The major conclusions of this study are summarized as follows.

- The aging-related bugs prediction models trained using decision tree (DT) have relatively better AUC values as compare to other techniques.
- The aging-related bugs prediction models developed using aging-related bugs (AGM) metrics as input have relatively better AUC values as compare to other sets of metrics.
- Upscale sampling technique provides relatively better performance in detecting aging-related bugs compared to the models built by original data.

## References

[1] Aggarwal, K., Singh, Y., Kaur, A., Malhotra, R., . Empirical analysis for investigating the effect of object-oriented metrics on fault proneness: a replicated case study. Software process: Improvement and practice .

[2] Chidamber, S.R., Darcy, D.P., Kemerer, C.F., 1998. Managerial use of metrics for Object-Oriented software: An exploratory analysis. IEEE Transactions on Software Engineering 24, 629–639.

[3] Chidamber, S.R., Kemerer, C.F., . Towards a metrics suite for Object-Oriented design. volume 26. ACM.

[4] Grottke, M., Li, L., Vaidyanathan, K., Trivedi, K.S., 2006. Analysis of software aging in a web server. IEEE Transactions on reliability 55, 411–420.

[5] Malhotra, R., Chug, A., 2014. Application of group method of data handling model for software maintainability prediction using object oriented systems. International Journal of System Assurance Engineering and Management 5, 165–173.

[6] Malhotra, R., Singh, Y., . On the applicability of machine learning techniques for object oriented software fault prediction. Software Engineering: An International Journal .

[7] Natella, R., 2017. Aging-related bugs and software complexity metrics Aging-related bugs and software complexity metrics. URL: https://doi.org/10.5281/zenodo.581659, doi:10.5281/zenodo.581659.

[8] Padhy, N., Singh, R., Satapathy, S.C., 2017. Enhanced evolutionary computing based artificial intelligence model for web-solutions software reusability estimation. Cluster Computing , 1–18.

[9] Sun, C., Wang, B., Liu, Y., Sun, D., Chen, D., 2016. Numerical simulations for near-field acoustic holographic data extrapolation based on the neural network elm method, in: Information Science and Electronic Engineering: Proceedings of the 3rd International Conference of Electronic Engineering and Information Science (ICEEIS 2016), January 4-5, 2016, Harbin, China, CRC Press. p. 87.

[10] Zheng, J., Okamura, H., Li, L., Dohi, T., 2017. A comprehensive evaluation of software rejuvenation policies for transaction systems with markovian arrivals. IEEE Transactions on Reliability 66, 1157–1177.