An Empirical Study of Aging Related Bug Prediction Using Cross Project in Cloud Oriented Software

Harguneet Kaur and Arvinder Kaur
University School of Information, Communication and Technology (USICT), GGSIPU, Dwarka, New Delhi, India
E-mail: harguneet.13316490018@ipu.ac.in, arvinder@ipu.ac.in

Keywords: Aging related bugs, cross project, cloud-oriented, empirical study, imbalance, software aging

Received: May 23, 2022

Cloud oriented applications enable users to increase the scalability of computing infrastructure by using shared computer resources. These applications include the characteristics such as NoSQL database, Big-Data Analytics, distributed file system and MapReduce architecture which may face issues like software aging due to which ongoing system's performance decreases and failure rate increases. Aging Related Bugs (ARB) are bugs that are caused due to memory leakage, null pointer exception, resource depletion etc. in the ongoing system whose impact can be dangerous, so it's better to predict them before releasing the software. Manual extraction of ARB reports are common but finding ARBs within thousand of bug reports is challenging. This is the first paper that presents the empirical study to automatically search aging related bug reports through SEARCH_KEYWORD algorithm and implement the ARB prediction in cross project for cloud oriented applications/softwares. To compare the efficacy of the prediction results, With-in Project Defect Prediction (WPDP) of ARBs is also performed. The work is divided in three phases: 1. ARB reports are extracted from the summary/description of bug in bug repository through automatic process. 2. Cross project bug prediction (CPDP) is performed to predict ARB due to limited availability of training data which is not implemented yet in cloud oriented softwares to the best of our knowledge. 3. Machine learning techniques are applied for ARB prediction to build fault prediction models. There is an imbalanced proportion between ARB-prone and ARB-free files, therefore Recall, FPR(False Positive Rate), Balance are used as major performance measures to predict ARBs. Kruskal Wallis Test and Friedman Test, are applied on the prediction results and it is proved that Naive Bayes performed significantly better than other classifiers. The results suggested that CPDP performed better than WPDP of ARBs using machine learning classifiers in cloud oriented datasets.

Povzetek: Narejena je empirična analiza napovedovanja programskih napak v oblakih.

1 Introduction

A Software bug is an error, fault or defect that has the capability to produce unexpected or inappropriate results. Software Bugs are classified into two types: Bohrbugs and MandelBugs. Bohrbugs are the bugs that are easily isolated and detected as failure and are easily reproducible whereas Mandelbugs are the bugs where the failure is not reproducible. ARB falls under the category of MandelBugs. ARBs are the errors in software at runtime which are caused due to memory leakage, resource depletion, null pointer exception, unreleased files and locks etc. It is challenging to locate aging related bugs in the software after its deployment, and thus fixing and reproducing them becomes a task. These bugs are caused due to the aggregation of errors in running software for a long period of time that can lead to performance degradation, depletion of resources, and in some cases, the software can even crash. Many open source softwares related to cloud computing environment are being used in recent years in which distributed applications are developed which use shared resources for the requirement of increased computing capability, memory, bandwidth and storage area. Some of the software's use cloud computing technology, for instance Hadoop MapReduce, Hadoop HDFS, Hive and Storm based on Mapreduce framework and Big data analytics. Cassandra has the unique functionality of NoSQL database of cloud computing technology. The cloud oriented softwares [1]are analysed by Machida et al. [2] to find the presence of aging related bugs. The results of his study confirmed the presence of ARBs in cloud oriented softwares [3].
Due to the behavior of ARB, it is difficult to detect and observe ARBs in the software before its released. ARB prediction is considered as an Independent problem from other defect predictions. ARBs are caused due to the accumulation of errors in long period of time and the effect of these can be deadly also. So it is difficult to reproduce and locate them that is why ARB prediction is taken independently in this research. Manual classification [4] of ARBs is not possible with large distributed and cloud oriented softwares; therefore the automatic process of extracting ARBs from thousand bug reports is proposed in this study. The bugs are extracted from the REST API of JIRA and then

aging related bug reports are extracted automatically using text mining by exploring the description of bug reports and finding aging related keywords in them such as memory, overflow, leakage, race condition etc. The automation part reduces time in searching ARBs in large datasets and the outcome produced is reliable and dependable also. ARBs exist in a very few number as compared to non-aging bugs which led to improper training set for bug prediction. Software bug prediction [5] is the process of predicting bug-prone areas to strengthen the reliability and quality of the software and minimize the cost and effort which might arise at the later stage for fixing the defects. In particular, the aim is to allow the developers to focus on the fault-proneness areas; therefore it is necessary to predict the location of bugs in files before the release of software so that more testing resources are focused on them to assure the quality of software. Bug prediction is performed to build the classification models from historical data using machine learning algorithms to predict bug-prone areas. If historical data is not available in sufficient amount then in such cases, Cross Project Defect Prediction (CPDP) is used. The CPDP methods attempt to construct a prediction model on one software project (training dataset) for sufficient past data and then use those learned trainers to predict the unknown labels of another project (testing dataset) [6][7]. In the literature, CPDP is implemented in datasets such as Linux, MySQL, Cardamom [8] but not in cloud oriented softwares. Issues faced so far is the manual classification of ARBs from all bug reports by reading the description of bug reports and cross project bug prediction which has not been implemented in cloud oriented softwares as per our knowledge. Therefore in this study, the automatic process of classifying ARBs through SEARCH_KEYWORD algorithm is proposed and cross project aging related bug prediction has been performed on cloud oriented softwares.

This paper focuses primarily on automatic extraction of ARBs from the bug repository in cloud oriented softwares to save time and cost during testing. It is an important issue when large systems like cloud oriented softwares are made up of thousands of modules. The proposed study uses software metrics as predictive variables and exploits machine learning classifiers using CPDP approach in cloud oriented softwares to build fault prediction models. Then these results are validated with the application of the statistical test. The number of ARB reports found in the dataset is less as compared to ARB-free bug reports therefore, these datasets are said to be imbalanced. These Aging related bug reports are mapped to respective java class files of datasets that whether each file is associated with ARB or not. Thus to handle class imbalance problem, CPDP is implemented on each cloud oriented software to predict ARBs. This study aims to compare the prediction results of the CPDP approach with WPDP of ARBs using machine learning classifiers [9].

The research paper is organized as follows: In Section 2, related work on the software aging problem is thoroughly discussed. Section 3 discusses the Framework for ARB prediction of cloud oriented software and research questions formulated based on our proposed work. Section 4 describes the research methodology including the Aging related keywords, software complexity metrics and machine learning classifiers used in this study. This section also explores the empirical research representing the cloud oriented datasets, Evaluation measures and experimental setup followed by section 5, where results and analysis are presented in detail. Section 6 reports the threats to validity and section 7 summarizes the results of our study along with the future direction.

2 Related work

Software aging depends on the life span of software where software grows older and starts showing failures (ARB) which degrade the performance and even lead to hanging or crashing the system. This study worked on prediction of aging related bugs in cloud oriented software. The work is segmented into three parts: 1. Extraction of aging related bug reports from thousand of bug reports using automatic process. 2. Implementing CPDP and WPDP approach for predicting ARBs using machine learning classifiers in cloud oriented softwares 3. Comparing the prediction results of CPDP and WPDP and to find out which classifier worked better. Table 1 focused on the objectives of the related studies, datasets used in various studies, the metrics used to achieve the purpose and the research gap in each study. ST1 and ST3 performed aging related bug prediction but it has manually sorted ARBs in bug reports. With large dataset when there are thousand bug reports, it is hard to sort it manually therefore automata is proposed in this study. Lov Kumar et al. [10] related source code metrics with ARB prediction and applied machine learning classifiers to predict aging related bugs in Linux, MySQL and ApacheHTTPD. The Software aging concept was investigated systematically by Huang in 1995.

Work presented by Cotroneo [11] is the empirical analysis on the relation between software aging and static metrics such as project size and complexity of the software. It stated that aging related bugs can be predicted with the help of Software complexity and size metrics and built the fault prediction models using the benchmark datasets i.e. MySQL, and Linux. Cotroneo et al. [8] predicted aging related bugs in three large software projects and then, with the help of code and Aging related metrics, built bug prediction models. But the limitation found in Cortroneo's research is that the bug reports are manually analyzed to identify ARB's in every software module. Cotroneo et al. [12] discussed the present status of the software aging and rejuvenation(SAR) community and highlighted the issues towards which research should be headed in the future. This process tends to cause the system's failure after running continuously for a considerable time due to changes in the software. Massimo Ficco [13] worked on real-time data processing applications like apache storm to predict software aging by identifying its symptoms to prevent the harmful and dangerous effects in the running environment. Xiaohui Wan et al. [14] analyzed that ARB-prone files are small compared to non ARB-prone files in software and remove this shortcoming by proposing a Supervised Representation Learning Approach (SRLA) technique based on a double encoding layer autoencoder in Linux, MySQL and Apache HTTPD. Software Bug prediction helps the testers to ensure software reliability by predicting fault-prone areas [15], [16], [17]. ST8 applied extreme learning machine with three different kernals and presented the aging related bug prediction results in Linux and MySQL. ST2, ST5 and ST6 are the studies where cross project bug prediction is implemented for aging related bugs using transfer learning approach on Linux, MySQL but not on cloud oriented softwares. ST4 confirmed the existence of ARBs in cloud computing datasets but has not performed prediction. Shraban Kumar Apat et al. [18] presented different approaches for defect prediction of the softwares and then applied machine learning techniques for them [19]. Therefore based on the above studies , the State of the Art (SOTA) in this study is sorting aging related bugs through automata in cloud oriented softwares and then predicting the bugs using cross project and within project approach with the help of machine learning classifiers. Table 1 includes the summary of the related work which describes the accuracies of each studied research article along with the research gap. As per the observation, the work done till now is manually exploring the aging related bugs from the bug reports using keywords but in our study it is done through automata where aging related bug reports are refined from thousands of bug reports using aging related keywords. As per our knowledge, cloud oriented datasets are not considered till now for predicting aging related bugs. But in this study cloud oriented softwares are evaluated. Cross project bug prediction approach is used in this study for better availability of training datatsets.

3 Framework for ARB prediction of cloud oriented software

Bug prediction of cloud oriented softwares undergoes three phases: Automatic approach for keywords, applying CPDP and WPDP approach for ARB prediction and then building prediction models in open source cloud oriented datasets. The findings obtained in this work through automata helps in the development of cross project bug prediction models. It is aimed to design the framework of ARB prediction shown in Figure 1 which define four modules of Aging related bug prediction: ARB automatic collection through scripting (M1), formulated dataset of cloud oriented softwares (M2), Cross project defect prediction (CPDP) of cloud oriented softwares (M3) and Within Project Defect Prediction in cloud oriented dataset(WPDP) (M4). The inspection is focused on the bug reports which are marked as CLOSED and RESOLVED. The bug reports which are fixed and unique (not duplicates of other reports) are considered in this study. Firstly, Bug reports are collected from the Jira repository and then automata is implemented by reading the description of bug reports for refining Aging related bug reports. Every Aging related bug report is searched with its bug id in GitHub and then the java class files effected by these bugs are labelled as ARB-prone. In the dataset, all java files are then classified as ARB prone files or non-ARB prone files with the help of labelled ARB metric. Software metrics and this labeled ARB metric are used to predict ARB bugs in cloud oriented dataset. CPDP is applied with machine learning algorithms on the cloud oriented datasets on the one hand and on the other side, WPDP is implemented to compare the results for predicting the bugs. The performance measures are evaluated to compare prediction models and statistical tests to validate the results. The novelty of this framework is in modules M1 and M3. This paper's main contribution is 1) According to our knowledge, this is the first empirical study performing cross project aging related bug prediction in cloud oriented softwares. 2) This study proposed extraction of aging related bug reports from other bug reports using keywords through an automatic process with the help of python.3) WPDP is also applied and then the results are compared with CPDP. Several fault prediction models are evaluated and compared to identify which classifier works better for ARB prediction. 4) Statistical tests such as Friedman and Kruskal Wallis Test are applied on prediction results for validating the hypothesis proposed in this study. The comparative study is done for cross project and within project defect prediction in cloud oriented datasets. Manual extraction of bug reports has been widely used in the past by reading the description or summary of each bug report and finding if aging related keyword is there. This study has reduced the time taking task by implementing SEARCH_KEYWORD algorithm shown in Algorithm 1 to extract aging related bug reports.The outcome produced in the algorithm is the aging related bug reports from thousand bug reports using their summary (description), issue id and issue key of each bug report found in cloud oriented datastet.

The empirical study of aging related bug prediction in cloud oriented softwares formulated the following research questions (RQs)

RQ 1. Software aging is an imbalanced data problem. How does it justify?

RQ 2. How is the prediction performance differentiated from CPDP with WPDP approach for ARB prediction in cloud oriented softwares ?

Table 1: Related Work

SerialNo.	Author	Objective	Dataset	Performance measure	Research Gap
ST1	Xiaoxue Wu [20]	It manually explored aging related bug reports in invalid bug reports and then the prediction is done to evaluate its performance	HDFS and HBase	Recall, Precision, F1-score and AUC	Manually explored ARBs in all bug reports
ST2	Qin, Fangyun, et al [21]	Proposed transfer learning approach for ARB prediction in cross project.	Linux, MySQL, Apache HTTPD	PD (probability of detection), PF (probability of false alarms), Bal (Balance)	Performed cross project ARB prediction not in cloud oriented softwares
ST3	Lov Kumar et al. [10]	It presented ARB prediction with the help of source code metrics using machine learning techniques	Linux, MySQL	Area under ROC , F-measure and accuracy	Manually explored ARBs in all the bug reports
ST4	Fumio Machida et al.[2]	It confirmed the existence of aging related bug reports in five cloud computing open source softwares	Hadoop Mapreduce, Cassandra, Eucalyptus , Memcached and Xen	Classification	It has confirmed the presence of ARB in cloud oriented softwares but did not perform prediction. It manually sorted ARBs in bug reports
ST5	Qin, Fangyun, et al.[22]	proposed the transfer learning aging related bug prediction approach (TLAP)	Linux and MySQL	Probability of Detection, PF and Bal	It performed cross project using TLAP on Linux and MySQL not on cloud oriented softwares
ST6	Fangyun Qin et al.[23]	presented the empirical study on cross-project ARB prediction in terms of Normalisation methods, Kernal functions and data mining techniques .	Linux, MySQL, HTTPD	PD, PF and Bal	It performed cross project using TLAP on Linux and MySQL not on cloud oriented softwares. It manually explored the ARBs in bug reports

ST7	Steffen Herbold [7]	It presented the systematic mapping study on cross project defect prediction	NASA, SOFT-LAB, JU-RECZKO, RELINK, AEEEM, MOCKUS, ECLIPSE, NETGENE, AUDI	Various performance measures depending on the availability of dataset	Cross project defect prediction is systematically explained
ST8	Lov Kumar et al.[24]	It applied extreme learning machine with three different kernals and presented the results for aging related bug prediction	Linux and MySQL	Accuracy , F-measure and AUC	Linux and MySQL datasets are used not cloud oriented softwares
ST9	Cotroneo D et al.[8]	Built fault prediction models for predicting source code files which are more prone to Aging-Related Bugs in software complex datasets.	Linux Kernal , MySQL DBMS, CARDAMOM middleware platform	Program size, Aging-Related Metrics (ARMs) , Halstead metrics, McCabe's cyclomatic complexity	Cross project defect prediction is not explored in detail
ST10	Tan, L. et al.[25]	Classified the bugs into various categories such as memory bugs, semantic bugs, or concurrency bugs	Linux kernel, Mozilla , Apache	Precision, Recall , F1	Classification of bugs has been done

RQ 3. Is there any statistical difference in machine learning algorithms using cross project for ARB prediction?
The existence of aging related bugs is confirmed in cloud computing datasets [2] but cross project bug prediction has not been performed. In this study, cross project and Within bug prediction, both are applied in the dataset to predict ARBs. Automatic extraction of aging related bugs from the thousand bug reports is performed based on aging-related keywords. The answers to these research questions provide good practice for researchers to enhance their prediction in a large bulk dataset, i.e. cloud oriented softwares.

Algorithm 1: SEARCH_ALGORITHM

```
Input: Import the Excel File containing the Bug
       Description, Bug Id of all the bug reports of
       1 Dataset
1.Create the list of all the bug reports containing
  Bug Description, Bug Id and Bug Key
2. Create the List of all the Aging Related
  Keywords-
'race', 'leak','memory','aging','overflow','deplet',
'Overflow','NPE', 'null pointer',
'Buffer exhausted', 'deadlock', 'flush',
  'Leak','Memory','LEAK',
'MEMORY','OVERFLOW', 'null pointer
  exception'
3. while Check if keywords are present in the list of
  all the bug reports do
    Create a new list of only aging related bug
      reports;
4. Remove the duplicates of aging related bugs if
  any in the list using the set conversion.
5. Convert the new list of aging related bug reports
  back to the dataframe and then to the excel file.
Output: Export the excel file containing only the
        aging related bug reports
```

Table 2: Keywords

Keywords			
Race	Leak	Memory	Aging
Overflow	deplet	overflow	NPE
null pointer	Buffer Exhausted	deadlock	flush
leak	LEAK	memory	MEMORY

4 Research methodology

4.1 Techniques revisited

4.1.1 Aging related keywords

Aging related bugs are classified by the effects of aging while using the resources for the long term. Some of the software resources have limited time and memory eg. data objects and threads, file descriptors and database connections; unreleased files and locks. When this limits extends it cause memory leakage which may lead to deadlock state sometimes or even the software can crash. In this study, aging related bugs are found with the help of keywords such as memory, leak, race, overflow, null pointer exception, deadlock, etc. All these keywords are related to aging. In most of the studies, the bug report description is manually studied and if aging related keywords are found in it, those bugs are categorized as aging related bugs. But as per today's scenario, when the amount of dataset has increased drastically, manual categorization takes a lot of time. Thus in this research, we have proposed the automatic process to categorize aging related bugs with scripting in the python programming language. The keywords used in this study are given in Table 2 [25] .

Table 3: Software Metrics

Type	Metrics
McCabe cyclomatic complexity	AvgCyclomatic, AvgCyclomaticModified, AvgCyclomaticStrict, AvgEssential, Cyclomatic, CyclomaticModified, Etc.
Program Size	AvgLine, AvgLineBlank, AvgLineCode, AvgLineComment, CountClassBase, CountClassDerived, CountDeclClass, CountDeclClassMethod, CountDeclClassVariable, CountDeclExecutableUnit, Etc.

Table 4: Dataset Description

Dataset	Version	Files	ARB-prone files
Cassandra	3	2764	385
Hadoop Mapreduce	0.23.0	1412	161
Hadoop HDFS	0.20.0	521	170
Hive	3.1.0	7112	367
Storm	2.3	2371	131

4.1.2 Software metrics

The objective of our study is to predict aging related bugs with the help of software metrics [26]. A software metric is an estimate of software characteristic which is measurable and countable. These metrics play an important role in measuring software productivity, performance, planning, and many other applications. The software metrics are summarized in Table 3, which have been automatically extracted using the Understand tool for static code analysis. Metrics are related to the bugs for fault prediction [11]. Our study used software metrics to predict aging related bugs. There are two sets of software metrics evaluated in our research:

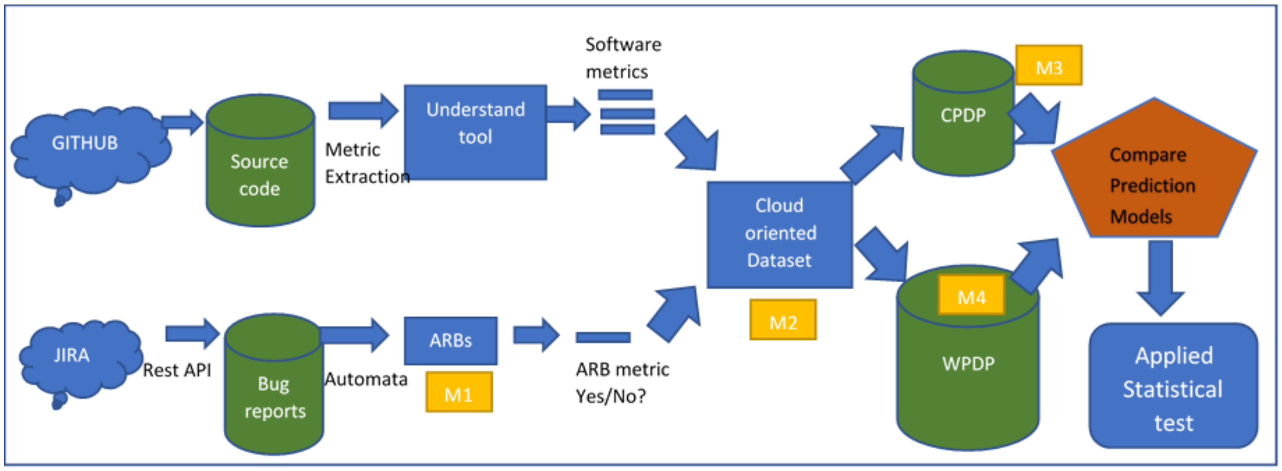

Figure 1: Framework for ARB prediction

Program size and McCabe's cyclomatic complexity. Program size is associated with the program's size in terms of lines of code and files and evaluates the developer's productivity. To measure the program's complexity, we require another set of metrics, i.e. McCabe's cyclomatic complexity. It controls the number of paths through a program in a graph and measures the number of operands and operators.

4.1.3 Machine learning classifiers/ classification models

Classification Models tries to pull out valuable conclusions from the observed values. In machine learning [27], two approaches can be followed: supervised and unsupervised. In supervised learning, the training dataset is provided into the classification algorithm, and then the testing dataset is compared with it to predict the outcome values whereas in unsupervised learning, the provided dataset is not labeled and looks for datapoints in the cluster. In unsupervised learning, the algorithm looks for similarities and patterns and identifies outliers in a dataset. Classification is an example of supervised learning and clustering is an example of unsupervised learning. The problem studied in this paper is a classification problem where it has to predict that software has bugs or not.
The relationship of software metrics and ARBs has been proved using machine learning algorithms widely applied in science to discover valuable patterns when a complex and large data volume is available [16] . The problem statement of predicting ARBs is supervised learning, also known as a binary classification problem. It predicts whether the data sample is ARB-free or ARB-prone in the testing dataset. In our research, the data sample mentioned is a java file called a module. Our study's focus is not to predict the number of aging related bugs in each file but the presence of bugs in each java file of our cloud oriented softwares. The number is not considered as these bugs are in a very small proportion and these bug's reproducibility is also difficult. Cross project defect prediction is performed in this study with cloud oriented softwares and then built the fault prediction models using general machine learning algorithms like Naive Bayes, Logistic Regression, Sequential Mininmal Optimization, Decision Table, Random Forest, J48 and bagging. In this section classification algorithms are discussed which are generally adopted by researchers and remarkably affect the efficacy of fault prediction[19] [28]. Some widely used classifiers in CPDP such as Nearest Neighbour, Decision Tree implemented with Aging related bugs bring unsatisfactory results therefore these classifiers are not considered in this study.

Naive Bayes (NB) This classifier is established on the basis of Bayes theorem with the features that are strongly assumed to be independent. It is highly scalable, generally used in text classification, requiring several variables to be linear in a learning problem. Bayes theorem evaluates conditional probability where problem which is to be classified with n features(independent) and represented by a vector $x = (x_1, x_2, ...x_n)$

$$\text{P}(C_k|x_1, x_2....x_n)$$

For each of K possible outcomes or classes, C_k.
When feature takes large number of values, then the Bayes theorem is decomposed into:

$$P(C_k|x) = p(C_k)p(x|C_k)/p(x)$$

Where $P(C_k|x)$ is a Posterior probability and $P(x|C_k)$ is a Likelihood probability. In this study, the posterior probability of hypothesis C_k is to be determined if a module is ARB-prone. The evidence x contains information that has been collected for prediction purposes. This evidence are software complexity metrics or attributes used to classify software. In this work, each metric x_i is assumed to be independent of each other where $x_j! = x_i$ then the posterior probability is given as:

$$P(C_k|x) = [\sum P(x_i|C_k)]P(C_k)/P(x)$$

Logistic Regression (LR) This is a supervised classification algorithm where y variable (output) take discrete values for a given set of features x(input). Linear regression predicts continuous dependent variable whereas logistic regression predicts categorical dependent variable. Logistic forms S-shaped curve which forms exponential growth taking values between 0 and 1.

$$y = e^(b_0 + b_1 * x_1 + b_2 * x_2 ..... + b_n * x_n)/(1 + e^(b_0 + b_1 * x_1 + b_2 * x_2 ..... + b_n * x_n))$$

where $x_1, x_2 ... x_n$ are the independent features and $b_0, b_1 ... b_n$ are parameters of the functions. It is applied for binary classification problems. Y measures the probability of the class and these parameters are adjusted to model the data according to our requirement. Logistic regression help in predicting the likelihood of aging prone files.

Sequential Minimal Optimization (SMO) SMO is an algorithm for the problems cause during the training of support vector machines. It is an iterative algorithm used for solving quadratic programming. It divides the problem into series of smaller sub-problems that are solved analytically.

Decision Table (DT) This is a machine learning technique made up of a hierarchical table where each node at a higher level is divided by adding additional attributes to form another table. It is a visual representation of data that decides on which conditions, what action is going to perform. A Decision table algorithm is used as a supervised machine learning algorithm. The Decision tree uses the C4.5 algorithm where questions are based on the attributes, i.e. complexity code metrics. In decision tree the algorithm splits the data iteratively until the classification error is not reduced further. The root and inner node mention metrics, and leaves show class labels. The Decision table is transformed easily into a decision tree.

Random Forest (RF) The main component of random forests is decision trees and operated as an ensemble. Random forest is widely used for prediction in machine learning algorithms. Each tree in a random forest spits out a class prediction and the class with maximum votes is used for model prediction. The fundamental concept behind random forest classifier is that many uncorrelated trees are grouped as a committee that surpasses any individual constituent model. When trees become together, the prediction result is more exact than individual trees because the error rate gets low as trees protect each other.

J48 In weka C4.5 algorithm is represented by J48. It is an iterative algorithm where training sample data initialize with the root, and then the partition is created of those whose link of the common attribute between sub-values is not there. J48 classification is based on decision tree where leaves and nodes represent class level. It uses continuous and categorical values. It also provides a technique of imputation which deals with missing values based on available data. J48 algorithm provides the replacement process of the subtree that decreases the error of classification after replacing the subtree with a leaf.

Bagging Bagging is an ensemble classifier that applies a classifier each on random subset of the original dataset and then collects their predictions to get a final prediction. It reduces the variance of decision tree by initiating randomization in its process and developing an ensemble. It uses the Bootstrap Aggregating algorithm for classification. Each classifier is trained on a sample training dataset with replacement such that the size of each sample equals the size of the actual training dataset.

4.2 Datasets

Aging related bugs are challenging to observe in software due to their less presence and difficulty in reproduction. This limits the availability of training data to develop fault prediction model. Our study has collected cloud oriented softwares that are large in scale and widely used in software commodities. Aging related bugs exist in open source cloud oriented softwares such as Hadoop Mapreduce, Cassandra, Hadoop Hdfs, Hive and Storm. Aging related bugs are less in number as comapre to non aging related bugs therefore it is a class imbalance problem.Therefore these datasets in this study are considered to be imbalance with the proof given in Table 4. To carry out the empirical study, cross project aging related bug prediction experiments are conducted on five large datasets. The version, number of files , ARB prone files and percentage of ARB prone files for each dataset are given in Table 4. Datasets are taken from GitHub.

Apache Cassandra (C) This is a popular open source management system with a wide column store, distributed and NoSQL database system. Basically, it provides high availability with fast support for clusters belonging to multiple data-centers. Tables are created , altered and dropped at run-time without blocking updates and queries. It is java based which is controlled via Java Management Extensions(JMX).

Hadoop MapReduce (HM) It is the programming model that generates big datasets based on a parallel, distributed clustering algorithm. MapReduce is composed of two procedures: 1. Map procedure which is filtering and sorting, 2. Reduce procedure that executes the summary function. It has also achieved high scalability and fault tolerance and has become a part of Apache Hadoop. It is basically a framework where large applications can be written to process huge data in reliable cluster manner.

Hadoop HDFS (Hadoop Distributed File System)-(HH) It manages distributed storage across nodes and follows Master-slave architecture. It is different from other distributed systems as it is fault-tolerant and divided into two components: Name node and n number of data nodes. Name node act as Master i.e. metadata and data nodes act as slaves. For huge data, files are stored in various machines. It is also available for parallel processing and streaming of file system data. It provides authentication and file permissions.

Hive (HI) It is a data warehouse system based on SQL for reading, writing and managing large datasets. JDBC driver and command-line tool are available for connecting Hive. It is an open source project control by volunteers at Apache foundation. It provides SQL abstraction to implement SQL-like queries under Java without the need for low level Java-API. Apache Hive is used and developed by Facebook and other companies like Netflix, Amazon Web Services.

Storm This is similar to Hadoop which became a standard for real-time processing system used to process large amount of data. Storm is different from Hadoop as it is stateless and can process tens of thousands of messages per second on cluster. Apache Storm is used by Twitter , NaviSite and Wego to process data in a distributed and fault tolerant environment. Storm is user-friendly and reliable that can be used by small companies as well as large software industries.

4.3 Evaluation measures

Aging related bug prediction is the imbalanced prediction problem where ARB-prone files share a small percentage as compared to non ARB prone files. Therefore it is not recommended to use accuracy, precision, F-measure which are used in many studies as these performance measures are poor indicators for imbalanced data.Thus we have used Recall , FPR(False Positive Rate) and Bal(Balance) [8] as performance measures. Classification models when tested, gives the output as a confusion matrix shown in Table 5 from which the performance measures are evaluated. We consider TP as True Positives, FP as False Positives, TN as True Negatives and FN as False Negatives in this study.TP denotes if the module is ARB-prone and it is correctly classified and the FN is if the module is ARB-prone but not correctly classified. Similarly, TN means that the module ARB-free is correctly classified, whereas FP incorrectly classified ARB-free modules.

Recall (TPR) Also known as sensitivity generally used for imbalanced classification to calculate the coverage of minor class. It is a measure that evaluates the number of correct positive predictions made out of all positive predictions. Basically, it indicates the number of missed positive predictions.

Table 5: Confusion Matrix

		Prediction class	
		ARB free	ARB prone
Actual Class	| ARB free	TN	FP
	| ARB prone	FN	TP

$$Recall = \frac{TP}{(TP + FN)} \tag{1}$$

FPR (False Positive Rate) also known as Fall-out indicating the probability of false alerts. It evaluates the negative classes incorrectly identified as positive classes from total number of negative classes.

$$FPR(FalsePositiveRate) = \frac{FP}{(FP + TN)} \tag{2}$$

Balance Bal (Balance) A tradeoff is needed between recall and FPR which are themselves comparable measures. Bal evaluates the euclidean distance between Recall and FPR.

$$Bal = 100 - \frac{\sqrt{(0 - FPR)^2 + (100 - Recall)^2}}{\sqrt{2}} \tag{3}$$

4.4 Experimental setup

Aging Related Bugs (ARB) are software bugs that have a severe impact on the system availability due to the aggregation of errors after long term software execution. ARBs [25] [8] can be listed as memory bugs, Unterminated Threads, Unreleased files and lock , and Disk Fragmentation . The categorization of aging related bugs from all the bug reports has been done manually in recent years by reading each bug report's description. If the bug report is found to be memory or aging related then it is marked as Aging related bug report. We have searched keywords such as memory leak, deadlock, deallocating, dereference, overflow buffer, lock, improper synchronization, null pointer exception, race condition, socket leak and uninitialized variables in the description of bug reports. In this study, we have transferred our manual task to automata through scripting because it becomes tough for the researcher to search for keywords in the description/summary of each bug report when there are thousands of bug reports in cloud oriented softwares. Manual searching is still acceptable when we are working with one dataset. But when we have many datasets and that too large, then it gets complicated and time taking that is why we have implemented the automata of searching the keywords in bug reports. The SEARCH_KEYWORD algorithm, already shown in Algorithm 1, is used to refine aging related bug reports from thousands of bug reports. To confirm automata results, sorting of aging related

bugs is also performed manually by reading bug description and then the results are compared as shown in Table 6. It is observed that the number of ARB collected manually and through automata are almost the same and comparable, therefore Bug reports through automata are further studied and explored. For example, in Cassandra , the aging related bugs were found to be 601 manually and 677 through automata. Similarly, in Hadoop Mapreduce ARBs manually collected are 192 and by applying automata, it came to 190. In Hadoop HDFS, ARBs explored manually are 377 and through automata, it is found as 344. In storm and Hive dataset , ARBs inspected manually are 82 and 517 respectively whereas ARBs inspected through automata are 84 and 601 respectively. Thus we have seen that a number of bugs investigated through manual and automata are comparable. There is no considerable difference between the number of ARBs and it is better to implement algorithm for finding ARBs in large dataset.
This section describes the experimental setup for the empirical study. The efficacy of ARB prediction is evaluated by training a classification model using training dataset and then using this model to classify other dataset known as testing dataset in order to correctly predict ARB-prone files of unseen data instances. ARB prone files are those java files which have confirmed the presence of ARBs with the help of bug reports. These datasets are obtained from cloud-oriented softwares by dividing it into training and testing set. The division is performed in two ways to evaluate the proposed approach. Firstly, fault prediction is evaluated when data about ARBs in the project under analysis is available i.e. ARB-prone files are predicted by training a classification model using data of the same project. In order to execute this way known as within project defect prediction (WPDP), the dataset of same cloud oriented software is split into training set compose of known ARB-prone files and testing set where ARB-prone files are to be predicted. The training set which is available to developers represents the historical data about ARBs of the dataset. Similar analysis is provided for cross-project defect prediction(CPDP) which is the second executed way where the test dataset is made up of a specific project and training dataset is composed of data from another cloud oriented software. The experiment is performed on five large cloud oriented softwares, thus there are 5*5 = 25 pairwise cases altogether for within and cross project prediction where each dataset performed cross project and within the project. We have applied 7 classifiers, calculated 3 performance measures on 5 large datasets thus this produces 7*3*25= 525 results which are fair enough to predict ARB. The bug reports of each dataset are collected from JIRA through Rest API where at each round 1000 bugs are collected, therefore to retrieve a greater number of bug reports we have to change the minimum and maximum value of bug reports and repeat the extraction rounds and then thousands of bug reports are collected for each dataset in excel file. Then automatic process is performed on bug reports to extract aging related bug reports with the help of keywords defined in Table 2 using the SEARCH_KEYWORD algorithm mentioned in section 3. The source code of all five open source cloud oriented datasets are collected from Github repository. Software metrics are used in this study for predictive purpose which are automatically extracted from Understand tool [8] for static code analysis. The dataset for prediction is composed of software metrics (independent variables) and label (dependent variable) which are fed into the weka tool [29] for implementation of CPDP and WPDP of ARBs using different machine learning classifiers. The label i.e. dependent variable is marked as '1' if the file is ARB-prone otherwise it is marked as '0'. The marking of the dependent variable is done manually by exploring each bug reports and listed java files associated with it. Then different performance measures are evaluated from the prediction results to build fault prediction models for cloud oriented datasets.

5 Results and analysis

In this section, the answers to all the research questions are presented in detail. The analysis of bug prediction is performed based on various performance measures evaluated by applying general mining algorithms and imbalance machine learning classifiers.

RQ 1. Software aging is an imbalanced data problem. How does it justify?
Software aging is the phenomenon of system failure after a long and continuous runtime which may cause due to the software's ongoing changes. Reinstalling or rebooting the software is a short term fix but is not a successful remedy for every software. Software aging is caused due to memory leakage, deadlock, memory bloating , unreleased file locks, numerical error and many more. The major hindrance of this phenomenon is that it is not reproducible; therefore these bugs are not easily repairable. Software aging may also lead to software crashes and degradation. ARBs are challenging to find in software, but it may cause significant harm to the software or humans. Thus it is necessary to predict aging related bugs in software.Table 6 displays the description of bug reports collected through JIRA repository where as in Table 7, description of Java files in each dataset is shown after the mapping of bug reports to java files. In Table 7, it is clearly visible that in each cloud oriented software, the percentage of ARB files from total java files is very less. For example, in Cassandra total number of java files are 2764, out of which only 385 are aging related java files which turn out to be 13.93% only. Hence, it is easily proved that software aging is the imbalanced data problem as each large dataset has less than 50% of predicted class i.e. aging related files.Similarly, in Hive aging related files only constitute 5% out of 7112 total number of files. If the predicted class in the dataset is less than 50% then that dataset is said to be imbalanced. It is significantly proved that software aging is the imbalance problem found in cloud oriented softwares. Due to the low probability of

ARB-prone files compared to ARB-free files, software aging is considered class imbalanced data problem. The key challenge of class imbalance is that the minority class left behind in front of the majority class. It is necessary to predict minority class i.e. ARB-prone files to assure software reliability.

Table 6: Comparison of ARB from manually with automata

S. No	Cloud Computing Dataset	Total No. of Bugs	ARB Manually collected	ARB through automata
1	Cassandra	7869	601	677
2	Hadoop Mapreduce	2541	192	190
3	Hadoop HDFS	5125	377	344
4	Storm	1316	82	84
5	Hive	9598	517	601

Table 7: Percentage of ARB files

Dataset	Total Java Files	Aging Related Java Files	Percentage
Hadoop HDFS	521	170	32.62956
Cassandra	2764	385	13.92909
Hadoop Mapreduce	1412	161	11.40227
Hive	7112	367	5.160292
Storm	2371	131	5.525095

RQ 2. How is the prediction performance differentiated from CPDP with WPDP approach for ARB prediction in cloud oriented softwares?

Software aging is the class imbalance problem where the prediction class i.e. ARB prone files, are in the minority compared to Aging-free files that fall under the majority class. When data is imbalance, it becomes difficult to predict bugs due to insufficient training dataset. In this study,two different paths are implemented : i) cross project defect prediction (CPDP) is carried out in cloud oriented softwares. ii) Within project defect prediction (WPDP) is implemented for five large cloud oriented softwares. In WPDP, original dataset which is used as training , the same is used for testing as well but in CPDP due to the unavailability of enough dataset, the dataset which is trained is different from the testing dataset. Weka performs WPDP by splitting the dataset in 10 cross fold validation.

The performance measures: Recall, FPR and Balance are calculated for all five datasets in all the combinations of training and testing datasets with cross project bug prediction[8] and within defect prediction. The main objective is to compare the prediction results for Aging related bugs in cloud oriented open source software with the help of CPDP and WPDP approach using machine learning classifiers. Table 8 shows the average results for within project defect prediction(WPDP) and cross project defect prediction(CPDP) of aging related bugs for all the five cloud oriented softwares. Classification machine learning algorithms are performed using the WEKA machine learning tool. On observing the data , it is clearly seen that CPDP performs much better than WPDP in most of the cases for ARB prediction. Thus it is significantly proved that for predicting aging related bugs, cross project is the better option than With-in project defect prediction using machine learning algorithms. Table 9 presents the results in terms of Recall performance measure for each classifier in 25 pairwise experimented datasets for ARB prediction.The above five experimented datasets in the table denotes the Within-project approach and rest 20 datasets represent cross project approach. The classification algorithms adopted in this study are Naïve Bayes, Logistic Regression, SMO, Decision Table, Random Forest , J48 and Bagging. The best prediction result i.e. higher recall value is highlighted yellow for each experimented dataset.The high recall value denotes ARB-prone files are correctly classified. The high value of Balance performance measure provdes the best tradeoff between Recall and FPR. The result observed is cross project ARB prediction performed better than within project ARB prediction where Naive Bayes predicts better results than other machine learning classifiers according to the statistics of Recall performance measure. As shown in Table 9,the recall value for Cassandra WPDP is 0.362 whereas max CPDP value is 0.42 when different testing dataset is considered for the same.SMO has performed worst on average in all the pairwise cases during prediction. The low performance of other classifiers is due to the negative effect of the less percentage of ARB-prone files in the cloud oriented datasets. Overall, Naïve Bayes is opted for best machine learning algorithm to predict ARB-prone files among cloud oriented softwares according to the statistics of recall performance measure.

Table 8: Results of Aging relate bug prediction in cloud oriented softwares

	Training	Cassandra			Hadoop Mapreduce			Hadoop HDFS			Hive			Storm		
Testing		Recall	FPR	Bal	Recall	FPR	Bal	Recall	FPR	Bal	Recall	FPR	Bal	Recall	FPR	Bal
Cassandra	NB	0.36	0.08	29.54	0.36	0.09	29.54	0.47	0.16	29.62	0.1	0.02	29.36	0.17	0.03	29.41
	LR	0.14	0.02	29.39	0.19	0.05	29.42	0.58	0.25	29.7	0.01	0	29.3	0.32	0.08	29.52
	SMO	0	0	29.29	0	0	29.29	0.08	0.01	29.35	0	0	29.29	0	0	29.29
	DT	0.11	0.02	29.36	0.1	0.02	29.36	0.43	0.13	29.59	0.01	0	29.3	0.16	0.03	29.41
	RF	0.14	0.03	29.39	0.06	0.01	29.34	0.39	0.1	29.57	0.01	0	29.3	0.08	0.01	29.34
	J48	0.25	0.07	29.47	0.17	0.05	29.41	0.47	0.2	29.62	0.02	0.01	29.3	0.18	0.04	29.42
	Bagging	0.16	0.03	29.4	0.05	0.01	29.33	0.45	0.13	29.61	0	0	29.29	0.09	0.02	29.35
Hadoop Mapreduce	NB	0.28	0.09	29.49	0.27	0.08	29.48	0.42	0.23	29.59	0.11	0.02	29.36	0.16	0.03	29.4
	LR	0.08	0.01	29.35	0.12	0.02	29.37	0.31	0.13	29.51	0.03	0	29.31	0.17	0.03	29.41
	SMO	0	0	29.29	0.01	0	29.3	0.12	0.02	29.37	0	0	29.29	0	0	29.29
	DT	0.11	0.02	29.36	0.08	0.01	29.35	0.31	0.13	29.51	0.02	0	29.3	0.14	0.04	29.39
	RF	0.12	0.02	29.37	0.21	0.01	29.44	0.36	0.13	29.54	0	0	29.29	0.1	0.02	29.36
	J48	0.24	0.11	29.46	0.24	0.04	29.46	0.42	0.23	29.59	0.05	0.01	29.32	0.18	0.05	29.42
	Bagging	0.16	0.03	29.4	0.21	0.02	29.44	0.41	0.16	29.58	0.01	0	29.3	0.14	0.03	29.39
Hadoop HDFS	NB	0.26	0.05	29.47	0.26	0.05	29.47	0.56	0.23	29.69	0.16	0.03	29.4	0.17	0.03	29.41
	LR	0.08	0.01	29.34	0.09	0.01	29.36	0.38	0.12	29.56	0.01	0	29.29	0.2	0.03	29.43
	SMO	0	0	29.29	0.01	0	29.29	0.14	0.03	29.39	0	0	29.29	0	0	29.29
	DT	0.09	0.01	29.35	0.06	0.01	29.33	0.42	0.11	29.59	0.03	0	29.31	0.14	0.01	29.39
	RF	0.11	0.01	29.36	0.08	0.01	29.34	0.51	0.12	29.65	0	0	29.29	0.11	0.01	29.36
	J48	0.28	0.09	29.49	0.17	0.05	29.41	0.51	0.23	29.65	0.04	0.01	29.32	0.2	0.03	29.43
	Bagging	0.13	0.01	29.38	0.08	0.01	29.34	0.5	0.14	29.64	0.01	0	29.29	0.15	0.01	29.39
Hive	NB	0.42	0.14	29.59	0.42	0.14	29.58	0.48	0.26	29.63	0.21	0.06	29.44	0.22	0.05	29.45
	LR	0.16	0.02	29.4	0.16	0.03	29.4	0.44	0.17	29.6	0.05	0	29.33	0.2	0.03	29.43
	SMO	0	0	29.29	0.03	0	29.31	0.19	0.05	29.42	0	0	29.29	0	0	29.29
	DT	0.15	0.04	29.4	0.13	0.02	29.38	0.46	0.14	29.61	0.02	0	29.3	0.18	0.03	29.42
	RF	0.18	0.04	29.42	0.11	0.02	29.37	0.4	0.15	29.57	0.05	0	29.33	0.14	0.02	29.39
	J48	0.26	0.13	29.47	0.16	0.04	29.4	0.41	0.21	29.58	0.1	0.01	29.36	0.2	0.04	29.43
	Bagging	0.23	0.04	29.45	0.1	0.02	29.36	0.46	0.18	29.61	0.03	0	29.31	0.18	0.02	29.41
Storm	NB	0.28	0.07	29.49	0.27	0.07	29.48	0.43	0.12	29.59	0.11	0.02	29.37	0.18	0.02	29.41
	LR	0.1	0.01	29.36	0.07	0	29.34	0.36	0.07	29.54	0.01	0	29.29	0.19	0.01	29.43
	SMO	0	0	29.29	0.02	0	29.3	0.1	0.02	29.36	0	0	29.29	0	0	29.29
	DT	0.18	0.03	29.41	0.08	0.01	29.35	0.32	0.09	29.51	0.02	0	29.3	0.15	0.01	29.4
	RF	0.11	0.01	29.37	0.05	0.01	29.33	0.32	0.07	29.51	0.01	0	29.29	0.12	0	29.37
	J48	0.2	0.07	29.43	0.13	0.01	29.38	0.31	0.15	29.51	0.03	0	29.31	0.14	0.01	29.39
	Bagging	0.15	0.02	29.39	0.05	0.01	29.33	0.37	0.09	29.55	0.01	0	29.29	0.08	0	29.35

Table 9: Recall Performance Measure for all experimented datasets

Training-Testing Dataset	NB	LR	SMO	DT	RF	J48	Bagging
C -C	0.362	0.138	0	0.107	0.141	0.25	0.159
HM-HM	0.269	0.119	0.013	0.082	0.207	0.244	0.207
HH-HH	0.563	0.385	0.137	0.421	0.509	0.515	0.503
HI-HI	0.208	0.052	0	0.022	0.055	0.099	0.033
Storm-Storm	0.177	0.193	0	0.154	0.116	0.139	0.085
C-HM	0.282	0.082	0	0.107	0.119	0.238	0.157
C-HH	0.261	0.077	0	0.089	0.107	0.285	0.131
C-HI	0.42	0.159	0.003	0.153	0.18	0.257	0.227
C-Storm	0.277	0.1	0	0.177	0.108	0.2	0.147
HM-C	0.359	0.19	0.003	0.097	0.065	0.169	0.055
HM-HH	0.261	0.095	0.006	0.06	0.077	0.166	0.077
HM-HI	0.417	0.161	0.03	0.131	0.109	0.159	0.104
HM-Storm	0.27	0.07	0.016	0.085	0.054	0.131	0.054
HH-C	0.473	0.577	0.084	0.429	0.395	0.468	0.452
HH-HM	0.419	0.307	0.119	0.313	0.357	0.419	0.407
HH-HI	0.483	0.436	0.191	0.456	0.404	0.406	0.456
HH-Storm	0.431	0.362	0.1	0.316	0.316	0.308	0.37
HI-C	0.104	0.011	0	0.011	0.011	0.021	0.003
HI-HM	0.107	0.025	0	0.019	0	0.05	0.013
HI-HH	0.16	0.006	0	0.03	0	0.042	0.006
HI-Storm	0.108	0.008	0	0.016	0.008	0.031	0.008
Storm-C	0.167	0.32	0	0.164	0.076	0.185	0.089
Storm-HM	0.157	0.169	0	0.138	0.1	0.182	0.138
Storm-HH	0.172	0.196	0	0.143	0.107	0.202	0.148
Storm-HI	0.224	0.197	0	0.183	0.142	0.197	0.178

RQ 3.Is there any statistical difference in machine learning algorithms using cross project for ARB prediction ?
The hypothesis test is performed to point out the best classifier to confirm the difference between classifiers statistically significant. In this study, non-parametric tests: Friedman and Kruskal Wallis Test are conducted based on Recall performance measure at a significance level of 0.05 i.e. 95% confidence level. The Kruskal-Wallis Test analyses the response of more than two levels of just one factor on the result whereas Friedman Test analyses the response of two factors on the experimental result. Kruskal Wallis Test is the nonparametric equivalent of One Way ANNOVA whereas Friedman Test is the nonparametric equivalent of Two Way ANNOVA. There are total of 25 combinations of dataset with seven machine learning classifiers. Both the tests are performed on the basis of ranking. The hypothesis formulated for this study are:
H_0: There is no significant difference between the machine learning classifiers
H_1 : There is a difference between the machine learning classifiers.
H_0 is null hypothesis and H_1 is alternate hypothesis. Rejecting H_0 means that there is a statistical difference among the classifiers i.e. p value (probability of statistic test) which is supposed to be lower than the significant level of 0.05. It does not tell us where that difference is among the classifiers. Both the tests are one-tailed test because it uses chi-square distribution to obtain a p-value. Under two conditions Null hypothesis can be rejected : first is if p value is less than the critical value and another is if Friedman test value (Q) or kruskal wallis Test value (K) is greater than the critical value evaluated by implementing these statistical tests.
After applying Friedman Test, p-value<<0.0001 which is lower than the significant level (0.05) using chi-square distribution and we have Q value i.e. friedman test value which turns out to be 105.484. Q value observed is greater than the critical value (12.592) shown in Figure 4. Thus null hypothesis is rejected on the basis of Q and p value shown in Table 10 as Friedman Test indicates that Q value should be greater than critical value to reject null hypothesis . This concludes that all seven general classifiers have not performed the same. Figure 3 points out that Naïve Bayes performs better for predicting aging related bugs than other machine learning classifiers with respect to Recall value.Kruskal Wallis Test is also applied to confirm the results for CPDP on Aging related bugs. K value (Kruskal Wallis Test value) is observed on the basis of sum of the ranks for each classifier shown in figure 6. The value of K(59.301) is much higher than chi-square distribution i.e. K(critical)=12.592 seen in Table 11, therefore null hypothesis is rejected. It confirms the presence of an alternate hypothesis that says seven general machine learning classifiers applied to predict aging related bugs are different. Figure 5 shows Naive Bayes perform better results than other machine learning classifiers. Finally, this study analyze the efficiency of Naive Bayes classifier in both the conditions for WPDP and CPDP in ARB prediction models.

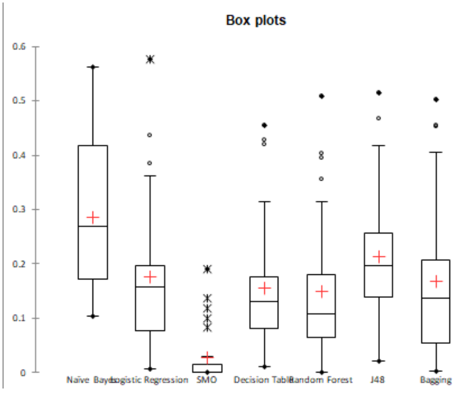

Figure 2: Distribution of classifiers on basis of Friedman Test

Table 10: Results of Friedman Test

Q (Observed value)	105.484
Q (Critical value)	12.592
DF	6
p-value (one-tailed)	0.0001
alpha	0.050

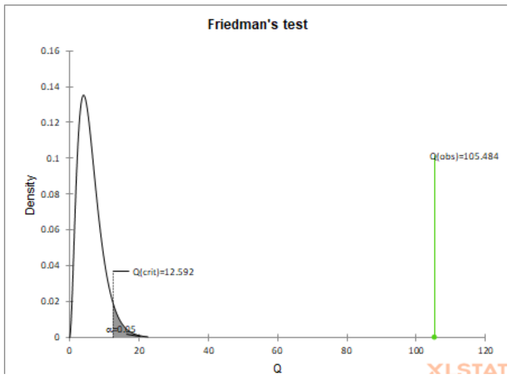

Figure 3: Graphical representation of Friedman Test

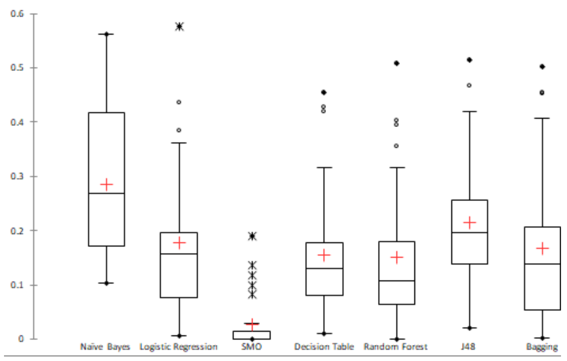

Figure 4: Distribution of classifiers on the basis of Kruskal Wallis Test

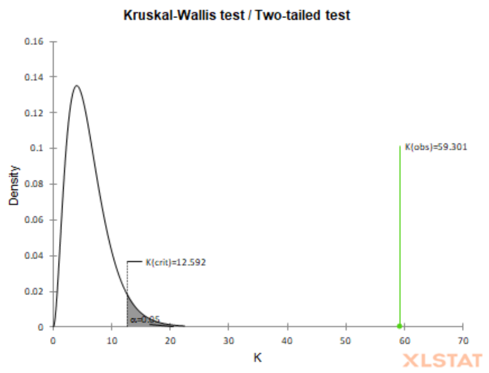

Figure 5: Graphical representation of Kruskal Wallis Test

Table 11: Results of Kruskal Wallis Test

K (Observed value)	59.301
K (Critical value)	12.592
DF	6
p-value (one-tailed)	0.0001
alpha	0.050

6 Threats to validity

In this section, limitations are discussed which are generally familiar in the empirical literature. Cloud oriented open source datasets have been taken for the experiment but it is suggested if industrial datasets are considered , it confirm the results and validate the proofs. In this study, software metrics are predicting aging related bugs in cloud oriented softwares but Aging related metrics and Halstead metrics are not extracted due to the industrial restrictions of Understand tool. It is observed in other studies [21] , aging related metrics are an vital component to predict aging related bugs for bug prediction models. Imbalance mitigation procedure and Transfer learning approach need to be applied for avoiding imbalance class problem. Cross project aging related bug prediction and WPDP is applied in five cloud oriented softwares with seven general machine learning classifiers that result in twenty-five datasets combined, still generalization cannot be done about the results as software is of different size with different development applications.

7 Conclusion and future work

An empirical study has been conducted to predict aging related bugs in cloud oriented software for building cross project bug prediction models. Five different cloud oriented datasets are explored with seven machine learning classifiers are considered in this study. Bug reports are extracted using JIRA REST API and then through automata approach, aging related bug reports are classified from all the bug reports based on the aging related keywords. Source code metrics of cloud oriented softwares are driven from Understand tool and then machine learning classifiers are applied to predict aging related bugs using cross project bug prediction and within project ARB prediction. The prediction results are then validated using the Kruskal Wallis Test and Friedman test where null hypothesis has been rejected and concluded that Naïve Bayes is the best among the other machine learning classifiers for predicting aging related bugs in the cloud oriented softwares. The following work has been contributed in our study:

1. Bug reports of cloud oriented softwares are in thousands, therefore it is not possible to directly download the bug reports thus we have used JIRA REST API in which at a time, every thousand bug reports is directly

downloaded in an excel file.

2. As per the research ,aging related bug reports are sorted manually by reading bug description on the basis of keywords in recent years but now when there are cloud oriented softwares containing thousands of bugs , it becomes tedious to do it manually. Hence we have created the algorithm and implemented in python to classify aging related bugs from all bug reports.

3. Software aging is a imbalance data problem where aging -prone files are in low proportion as compared to aging-free files. Cross project bug prediction is applied on five large datasets to predict the aging -prone class files for building bug prediction models.

4. Seven machine learning classifiers are executed on five different projects and CPDP and WPDP is carried out to predict ARBs. Naïve Bayes is proved to be the best among these classifiers to predict aging related bugs where it is proved that CPDP performed better than WPDP using general machine learning classifiers.

5. Non-parametric test i.e. Kruskal Wallis Test and Friedman Test are performed to validate the results. On application of these statistical tests, null hypothesis is rejected and it proved that all classifiers work differently.

In the future, it is planned to consider aging related metrics , Halstead metrics along with software metrics for predicting ARBs. Feature selection and imbalance mitigation procedures are going to be applied in future work. By using imbalance class methods, the results can be more generalized and confirmed as ARBs is the imbalance problem.Imbalance techniques are planned to be implemented in future work. More advance imbalance machine learning classifiers are planned to implement in cloud oriented datasets which will be enhanced with more experimentation on large size datasets.

References

[1] J. Araujo, R. Matos, P. Maciel, and R. Matias, "Software aging issues on the eucalyptus cloud computing infrastructure," in *2011 IEEE international conference on systems, man, and cybernetics*. IEEE, 2011, pp. 1411–1416, doi: 10.1109/icsmc.2011.6083867.

[2] F. Machida, J. Xiang, K. Tadano, and Y. Maeno, "Aging-related bugs in cloud computing software," in *2012 IEEE 23rd international symposium on software reliability engineering workshops*. IEEE, 2012, pp. 287–292, doi: 10.1109/issrew.2012.97.

[3] A. Bădică and Z. Z. Budimac, "Introduction to the special issue on "engineering and applications of software agents"," *Informatica*, vol. 40, no. 1, 2016.

[4] M. Ohira, Y. Kashiwa, Y. Yamatani, H. Yoshiyuki, Y. Maeda, N. Limsettho, K. Fujino, H. Hata, A. Ihara, and K. Matsumoto, "A dataset of high impact bugs: Manually-classified issue reports," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. IEEE, 2015, pp. 518–521, doi: 10.1109/msr.2015.78.

[5] M. D'Ambros, M. Lanza, and R. Robbes, "An extensive comparison of bug prediction approaches," in *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*. IEEE, 2010, pp. 31–41, doi: 10.1109/msr.2010.5463279.

[6] S. Herbold, "Training data selection for cross-project defect prediction," in *Proceedings of the 9th international conference on predictive models in software engineering*, 2013, pp. 1–10, doi: 10.1145/2499393.2499395.

[7] ——, "A systematic mapping study on cross-project defect prediction," *arXiv preprint arXiv:1705.06429*, 2017.

[8] D. Cotroneo, R. Natella, and R. Pietrantuono, "Predicting aging-related bugs using software complexity metrics," *Performance Evaluation*, vol. 70, no. 3, pp. 163–178, 2013, doi: 10.1016/j.peva.2012.09.004.

[9] H. Karna, S. Gotovac, and L. Vicković, "Data mining approach to effort modeling on agile software projects," *Informatica*, vol. 44, no. 2, 2020, doi: 10.31449/inf.v44i2.2759.

[10] L. Kumar and A. Sureka, "Feature selection techniques to counter class imbalance problem for aging related bug prediction: aging related bug prediction," in *Proceedings of the 11th innovations in software engineering conference*, 2018, pp. 1–11, doi: 10.1145/3172871.3172872.

[11] D. Cotroneo, R. Natella, and R. Pietrantuono, "Is software aging related to software metrics?" in *2010 IEEE Second international workshop on software aging and rejuvenation*. IEEE, 2010, pp. 1–6, doi: 10.1109/wosar.2010.5722096.

[12] D. Cotroneo, R. Natella, R. Pietrantuono, and S. Russo, "A survey of software aging and rejuvenation studies," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 10, no. 1, pp. 1–34, 2014, doi: 10.1145/2539117.

[13] M. Ficco, R. Pietrantuono, and S. Russo, "Aging-related performance anomalies in the apache storm stream processing system," *Future Generation Computer Systems*, vol. 86, pp. 975–994, 2018, doi: 10.1016/j.future.2017.08.051.

[14] X. Wan, Z. Zheng, F. Qin, Y. Qiao, and K. S. Trivedi, "Supervised representation learning approach for

cross-project aging-related bug prediction," in *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2019, pp. 163–172, doi: 10.1109/issre.2019.00025.

[15] S. S. Rathore and S. Kumar, "An empirical study of some software fault prediction techniques for the number of faults prediction," *Soft Computing*, vol. 21, no. 24, pp. 7417–7434, 2017, doi: 10.1007/s00500-016-2284-x.

[16] S. K. Pandey, R. B. Mishra, and A. K. Triphathi, "Software bug prediction prototype using bayesian network classifier: A comprehensive model," *Procedia computer science*, vol. 132, pp. 1412–1421, 2018, doi: 10.1016/j.procs.2018.05.071.

[17] R. Ferenc, "Bug forecast: A method for automatic bug prediction," in *International Conference on Advanced Software Engineering and Its Applications*. Springer, 2010, pp. 283–295, doi: 10.1007/978-3-642-17578-728.

[18] S. K. Apat, S. A. Rao, and P. S. K. Patra, "Software bug prediction analysis using various machine learning approaches," 2020, doi: 10.14569/ijacsa.2018.090212.

[19] A. Kaur and I. Kaur, "An empirical evaluation of classification algorithms for fault prediction in open source projects," *Journal of King Saud University-Computer and Information Sciences*, vol. 30, no. 1, pp. 2–17, 2018, doi: 10.1016/j.jksuci.2016.04.002.

[20] X. Wu, W. Zheng, M. Pu, J. Chen, and D. Mu, "Invalid bug reports complicate the software aging situation," *Software Quality Journal*, pp. 1–26, 2020, doi: 10.1007/s11219-019-09481-2.

[21] F. Qin, Z. Zheng, Y. Qiao, and K. S. Trivedi, "Studying aging-related bug prediction using cross-project models," *IEEE Transactions on Reliability*, vol. 68, no. 3, pp. 1134–1153, 2018, doi: 10.1109/tr.2018.2864960.

[22] F. Qin, Z. Zheng, C. Bai, Y. Qiao, Z. Zhang, and C. Chen, "Cross-project aging related bug prediction," in *2015 IEEE International conference on software quality, reliability and security*. IEEE, 2015, pp. 43–48, doi: 10.1109/qrs.2015.17.

[23] F. Qin, X. Wan, and B. Yin, "An empirical study of factors affecting cross-project aging-related bug prediction with tlap," *Software Quality Journal*, pp. 1–28, 2019, doi: 10.1007/s11219-019-09460-7.

[24] L. Kumar and A. Sureka, "Aging related bug prediction using extreme learning machines," in *2017 14th IEEE India Council International Conference (INDICON)*. IEEE, 2017, pp. 1–6, doi: 10.1109/indicon.2017.8487925.

[25] L. Tan, C. Liu, Z. Li, X. Wang, Y. Zhou, and C. Zhai, "Bug characteristics in open source software," *Empirical software engineering*, vol. 19, no. 6, pp. 1665–1705, 2014, doi: 10.1007/s10664-013-9258-8.

[26] A. Rathee and J. K. Chhabra, "Extraction and evaluation of software components from object-oriented artifacts," *Informatica*, vol. 45, no. 1, 2021, doi: 10.31449/inf.v45i1.3464.

[27] I. E. Araar and H. Seridi, "Software features extraction from object-oriented source code using an overlapping clustering approach," *Informatica*, vol. 40, no. 2, 2016.

[28] A. Ahmad, "Predicting software aging related bugs from imbalanced datasets by using data mining techniques."

[29] Z. Markov and I. Russell, "An introduction to the weka data mining system," *ACM SIGCSE Bulletin*, vol. 38, no. 3, pp. 367–368, 2006, doi: 10.1145/1140123.1140127.