Aging Related Bug Prediction using Extreme Learning Machines

Lov Kumar
NIT Rourkela, India
lovkumar505@gmail.com

Ashish Sureka
Ashoka University, India
ashish.sureka@ashoka.edu.in

***Abstract*—Aging-Related Bugs (ARBs) occur in long running systems due to error conditions caused because of accumulation of problems such as memory leakage or unreleased files and locks. Aging-Related Bugs are hard to discover during software testing and also challenging to replicate. Automatic identification and prediction of aging related fault-prone files and classes in an object oriented system can help the software quality assurance team to optimize their testing efforts. In this paper, we present a study on the application of static source code metrics and machine learning techniques to predict aging related bugs. We conduct a series of experiments on publicly available dataset from two large open-source software systems: Linux and MySQL. Class imbalance and high dimensionality are the two main technical challenges in building effective predictors for aging related bugs. We investigate the application of five different feature selection techniques (OneR, Information Gain, Gain Ratio, RELEIF and Symmetric Uncertainty) for dimensionality reduction and SMOTE method to counter the effect of class imbalance in our proposed machine learning based solution approach. We apply Extreme Learning Machines (ELM) with three different kernels (linear, polynomial and RBF) and present experimental results which demonstarte the effectiveness of our approach.**

***Index Terms*—Aging Related Bugs, Extreme Learning Machines, Imbalance Learning, Empirical Software Engineering and Measurement, Feature Selection and Extraction Techniques, Machine Learning, Predictive Modeling, Software Maintenance**

I. Research Motivation and Aim

Aging-Related Bugs (ARBs) are defined as those bugs which are caused due to error-conditions associated with software aging. An example of ARB is a system crash or hang due to gradual depletion of resources (such as memory leak in a web server) caused because of a long running system [1][2][3]. ARB happens when the execution time or period of the system is long as faults due to error conditions like memory leakage or unreleased files and locks gets accumulated over a period of time before a failure is triggered [1][2][3]. It is hard to uncover aging related bugs during system testing and also it is non-trivial to replicate such bugs. Early prediction of such bugs and identification of files or classes in a software system which are prone to aging related faults can help a quality assurance team in optimizing their testing efforts and resources. Recent research shows that static source code metrics can be used as predictors for aging related bugs within a machine learning framework [1][2][3]. However there are *two main technical challenges* in building predictive models for gaining related bugs.

Imbalanced Data: One of the technical challenges in building an ARB classifier is that the training dataset is highly imbalanced or skewed. In a binary classification problem (for example agining related bug prediction), the class of interest is in minority which poses technical challenges is building an effective classifier. For example, in a binary class problem, if the class of interest (aging related bugs) has only 1% instances in comparison to 99% instances of the majority class then the dataset is considered to be highly imbalanced.

High-Dimensional Data Research shows that a high-dimensional feature space consisting of irrelevant and redundant features decreases the performance of the classifiers [4][5][6][7][8]. The presence of large number of features (in our case, static source code features) poses intrinsic challenges to machine learning classification algorithms. Dimensionality reduction and identification of important features is a technical challenge in the context of ARB prediction.

Which classes or files in a given object oriented software system are fault-prone from the perspective of aging related issue or likely to contain gaining related defect? is a problem encountered by software practitioners and the work presented in this paper is motivated by the need to build tool support for software developers for identifying aging related fault-prone classes. There are lack of statistical analysis based studies examining the effectiveness of various dimensionality or feature reduction technique, techniques to counter the class imbalance problem and the application of extreme machine learning based method in the domain of aging related bug prediction using source code metrics. We frame three research questions aimed at filling some of the research gaps. While the authors of this paper have explored the problem of ARB prediction (a work under review), the application of ELM is novel and unique. The work presented in this paper is paper is guided by the following research questions:

RQ 1: Is there a statistically significant difference in the prediction performance of five different feature ranking techniques? Which feature ranking technique(s) yields the

best result?

RQ 2: Is there a statistically significant difference in the prediction performance of the predictive model after applying SMOTE method to address the class imbalance problem?

RQ 3: What is the relative performance of the Extreme Learning Machines (ELM) with three different types of kernels in-terms of the AUC, accuracy and f-measure metrics? Is there a statistically significant difference in the prediction performance of classifiers based on three different ELM kernels?

II. Experimental Dataset

We use a publicly available dataset available at tera-PROMISE Repository[1] for our experiments. The tera-PROMISE website is a well-known repository consisting of several software engineering research datasets on code analysis, defects, effort, refactoring and test generation [9]. We use a dataset from the tera-PROMISE repository so that our experiments can be easily replicated and can be used for benchmarking and comparison by other researchers. We conduct experiments on a dataset containing information on aging-related bugs for two large, complex and long-living software systems. The two open-source projects are the Linux kernel and the MySQL DBMS. The dataset by Roberto et al. [10] has been used in the past for building aging-related defect prediction models using software complexity metrics as features in a machine learning framework [3]. The dataset contains several types of metrics (82 metrics - refer to Table II) such as program size related metrics, McCabe's Cyclomatic complexity, Halstead metrics and aging-related metrics [10]. The metrics are independent variables and the number of aging-related bugs found in each file is the dependent variable.

Table I displays the experimental dataset details. The dataset consists of seven different projects. Four projects are Linux sub-systems and three projects are MySQL sub-systems. The largest of the seven projects is Linux driver net consisting of 2292 records out of which 9 records (0.39%) are ARBs. The smallest project in the dataset consists of 29 records in which the percentage of the minority class is 17.24%. Table I reveals that the dataset for all the projects is highly imbalanced. The percentage of minority class varies from 0.39% to 17.24%.

III. Research Framework and Proposed Technique

Our research framework consisting of a multi-step process. There are four main steps in the proposed approach and framework. The four steps are: (1) Feature ranking and selection (2) Learning from imbalanced dataset (3) Predictive model building using machine learning classifiers (4) Classifier evaluation using performance metrics and conducting statistical significance test. Each of the four steps are described below in detail. Furthermore, we follow a research methodology in which we formulate *three grounded research questions* which guides our study. We apply the Sandberg et al. approach of formulating research questions based on existing literature and theories [11].

Feature Ranking and Selection: We apply feature ranking and attribute selection to reduce the dimensionality of the input software metrics. Feature ranking is a pre-processing step applied before training or building a predictive model. The primary purpose of attribute ranking and subset selection is to reduce the number of redundant and irrelevant features and noise in the data. Feature selection improves the performance of the classifier and learning algorithm in-terms of speed, storage requirement and predictive accuracy. Less number of relevant and important features also makes the result easy to interpret and helps in understanding the target concept. We apply a feature-ranking approach in which every feature is ranked and then we define a criteria to select the top k ranked features. The higher ranked features have better prediction capability than the features which are lower ranked. The value of k in the top k feature is pre-defined and finally a subset of features are selected based on the k value and feature ranks.

There are two approaches for feature selection: filter based and wrapper based approaches. Filter based approaches consists of applying correlation coefficient between variables for ranking whereas wrapper based approaches assesses the usefulness of features according to a given predictor or learning algorithm [4]. Liu et al. present a survey of several feature selection algorithms [6]. We apply five different feature selection approaches: OneR, Information Gain (IG), Gain Ratio (GR), RELEIF (RF) and Symmetric Uncertainty (SU). We also consider no feature ranking or subset selection (all metrics or features). The selected features are shown in the form of shaded cells in the result Tables.

Learning from Imbalanced Dataset: The issue of class imbalance can be addressed through two approaches: data-level and classification algorithm level. The data level approach consists of oversampling the rare or minority class (the class which is of interest) or under-sampling the majority class. The oversampling or under-sampling

[1] http://openscience.us/repo/

TABLE I: Experimental Dataset Description

ID	Name	Total Records	Majority Class	Minority Class	% Majority Class	% Minority Class
Proj1	**Linux driver net**	2292	2283	9	99.61	0.39
Proj2	**Linux driver scsi**	962	958	4	99.58	0.42
Proj3	**Linux ext3**	29	24	5	82.76	17.24
Proj4	**Linux ipv4**	117	115	2	98.29	1.71
Proj5	**MySQL innodb**	402	370	32	92.04	7.96
Proj6	**MySQL optimizer**	36	33	3	91.67	8.33
Proj7	**MySQL replication**	32	28	4	87.5	12.5

TABLE II: 82 Features or Source Code Metrics Ranked by the OneR and Information Gain Feature Selection Technique for Seven Different Software Sub-Systems

	OneR							Information Gain						
	Proj1	**Proj2**	**Proj3**	**Proj4**	**Proj5**	**Proj6**	**Proj7**	**Proj1**	**Proj2**	**Proj3**	**Proj4**	**Proj5**	**Proj6**	**Proj7**
AltAvgLineBlank	82	82	72	81	4	72	39	37	82	82	82	34	10	82
AltAvgLineCode	30	30	33	30	53	40	50	54	30	35	30	44	33	50
AltAvgLineComment	29	29	32	29	40	56	48	39	1	34	29	47	32	49
AltCountLineBlank	28	28	77	28	66	57	69	3	29	32	28	1	30	24
AltCountLineCode	27	27	31	27	37	68	81	32	27	29	27	10	27	34
AltCountLineComment	26	26	79	26	78	65	65	9	24	31	26	35	29	48
AvgCyclomatic	23	23	30	23	35	81	66	58	26	30	23	39	8	51
AvgCyclomaticModified	24	24	29	24	36	66	23	40	25	36	24	42	28	45
AvgCyclomaticStrict	31	31	34	31	43	70	24	59	31	33	31	40	7	53
AvgEssential	32	32	28	32	44	80	14	51	28	37	32	43	3	56
AvgLine	33	33	35	33	60	42	15	1	32	43	33	41	34	55
AvgLineBlank	38	38	37	38	3	71	31	36	38	45	38	29	9	54
AvgLineCode	40	40	42	40	8	78	58	50	40	44	40	36	35	47
AvgLineComment	39	39	41	39	5	58	61	38	39	42	39	48	36	46
CountClassBase	37	37	40	37	12	45	22	42	37	39	37	54	37	42
CountClassCoupled	34	34	39	34	31	46	21	43	34	27	34	55	42	41
CountClassDerived	36	36	26	36	22	47	19	45	36	40	36	64	41	40
CountDeclClass	35	35	38	35	49	14	18	49	35	38	35	65	40	38
CountDeclClassMethod	25	25	36	25	21	7	33	48	33	28	25	66	39	43
CountDeclClassVariable	21	21	27	21	20	15	32	47	22	24	21	75	25	39
CountDeclFunction	4	4	76	4	55	16	71	24	13	26	4	9	31	31
CountDeclInstanceMethod	20	20	24	20	23	27	13	57	21	18	20	78	26	44
CountDeclInstanceVariable	9	9	25	9	19	28	27	61	10	17	9	79	21	57
CountDeclInstanceVariablePrivate	8	8	13	8	24	32	30	73	9	14	8	82	12	52
CountDeclInstanceVariableProtected	7	7	12	7	26	51	29	75	7	13	7	81	13	59
CountDeclInstanceVariablePublic	6	6	14	6	29	22	38	74	2	19	6	76	17	60
CountDeclMethod	5	5	15	5	28	20	37	77	6	16	5	74	18	74
CountDeclMethodAll	2	2	17	2	27	18	36	78	3	22	2	67	19	73
CountDeclMethodConst	3	3	22	3	25	19	35	79	5	23	3	73	20	76
CountDeclMethodFriend	10	10	23	10	17	23	28	80	11	15	10	68	23	75
CountDeclMethodPrivate	11	11	21	11	10	24	26	81	8	21	11	69	24	77
CountDeclMethodProtected	12	12	18	12	16	25	20	76	12	25	12	70	11	81
CountDeclMethodPublic	17	17	20	17	7	26	25	72	18	20	17	71	15	80
CountInput	19	19	19	19	6	29	17	82	20	46	19	72	22	79
CountLine	18	18	9	18	76	67	80	25	19	41	18	3	14	35
CountLineBlank	16	16	74	16	69	62	78	4	17	48	16	2	16	23
CountLineCode	13	13	10	13	50	21	6	29	14	66	13	7	43	17
CountLineCodeDecl	15	15	43	15	58	64	16	33	16	72	15	49	38	7
CountLineCodeExe	14	14	8	14	56	10	3	28	15	71	14	5	45	4
CountLineComment	41	41	45	41	79	74	64	8	41	12	41	8	55	26
CountLineInactive	42	42	44	42	68	5	76	14	42	47	42	38	82	58
CountLinePreprocessor	43	43	80	43	9	6	79	35	43	70	43	45	72	29
CountOutput	46	46	66	53	14	4	34	70	46	69	46	62	71	78
CountPath	71	71	64	70	15	3	40	62	71	68	71	63	69	72
CountSemicolon	70	70	68	69	63	8	11	18	70	6	70	6	66	2
CountStmt	68	68	7	67	57	9	9	22	68	73	68	4	68	9
CountStmtDecl	65	65	62	64	13	11	82	27	65	8	65	18	67	33
CountStmtEmpty	67	67	63	66	11	73	1	44	67	67	67	46	73	10
CountStmtExe	66	66	5	65	67	13	8	21	66	74	66	23	74	18
Cyclomatic	72	72	67	71	30	12	57	69	72	76	72	80	75	71
CyclomaticModified	69	69	65	72	18	30	56	68	69	81	69	60	76	70
CyclomaticStrict	73	73	69	73	32	17	59	67	73	80	73	61	81	63
Essential	79	79	71	76	33	31	60	66	79	79	79	57	80	62
Knots	81	81	73	80	46	43	55	63	81	78	81	53	79	61
MaxCyclomatic	80	80	70	79	54	79	75	64	80	77	80	16	2	30
MaxCyclomaticModified	78	78	61	78	71	1	72	71	78	75	78	14	5	25
MaxCyclomaticStrict	75	75	53	77	59	76	63	65	75	65	75	15	1	36
MaxEssentialKnots	77	77	59	74	47	49	62	60	77	57	77	56	78	64
MaxInheritanceTree	76	76	60	75	45	50	54	46	76	64	76	51	77	65
MaxNesting	74	74	50	68	48	52	47	52	74	54	74	52	70	66
MinEssentialKnots	64	64	48	63	38	55	53	56	64	53	64	58	65	69
PercentLackOfCohesion	63	63	46	62	42	54	52	53	63	52	63	77	64	68
RatioCommentToCode	62	62	47	61	34	53	44	55	62	51	62	28	63	67
SumCyclomatic	51	51	4	50	81	48	4	10	51	50	51	20	52	6
SumCyclomaticModified	50	50	11	49	70	77	5	5	50	55	50	19	51	3
SumCyclomaticStrict	49	49	2	47	77	33	10	12	49	49	49	21	49	5
SumEssential	48	48	51	44	75	44	43	26	48	3	48	13	46	11
n1	47	47	49	46	65	36	67	2	47	4	47	17	48	22
n2	44	44	52	45	64	35	7	20	44	10	44	24	47	12
N1	45	45	1	51	51	34	42	16	45	56	45	32	53	8
N2	52	52	57	48	1	37	45	17	52	9	52	31	50	13
Len	53	53	3	52	52	38	41	15	53	58	53	33	54	20
Voc	54	54	58	58	41	61	12	11	54	7	54	22	60	21
Vol	59	59	6	60	74	2	46	19	59	63	59	30	6	19
Dif	61	61	56	82	72	39	2	7	61	11	61	26	62	16
Eff	60	60	54	59	73	82	74	13	60	62	60	37	61	28
AllocOps	58	58	55	57	2	60	68	41	58	61	58	50	59	37
DeallocOps	55	55	78	54	39	41	51	34	55	60	55	59	56	14
DerefUse	57	57	75	56	62	69	73	31	57	59	57	25	58	1
UniqueDerefUse	56	56	81	55	61	63	49	30	56	2	56	27	57	15
DerefSet	22	22	82	22	80	75	77	23	23	1	22	12	4	32
UniqueDerefSet	1	1	16	1	82	59	70	6	4	5	1	11	44	27

can be random or informed. An informed over-sampling or under-sampling is not random and rather guided by a heuristic or criteria [12][13]. Oversampling and under-sampling can also be combined or blended. We apply data-level SMOTE technqiue to address the class imbalance problem. SMOTE (Synthetic Minority Over-sampling Technique) algorithm blends under-sampling of the majority class with a special form of over-sampling the minority class [14].

Learning Algorithms: We apply Extreme Learning Machines (ELM) with three different types of kernels (linear, polynomial and RBF). Extreme Learning Machines (ELMs) are a type of feed-forward neural network which can be applied for classification as well as regression problems [15]. On of the properties of ELMs is that they contain a single layer of hidden nodes [15]. ELMs can be used with various types of kernel functions.

Performance Evaluation: There are several different performance metrics for measure the effectiveness of a classifier. Sokolova present a systematic analysis of the performance measures for classification task [16]. We use Area under the ROC (Receiver Operating Characteristics) Curve (AUC), f-measure and accuracy to measure classifier performance. We use f-measure as it incorporates both precision and recall. Similarly, we use AUC as AUC has an attractive property of being insensitive to changes in the class distribution [16]. We apply cross validation technique to remove bias while evaluating the predictive models.

IV. Experimental Results

Feature Ranking: Table II displays the output of OneR and Information Gain (IG) feature ranking technique. We select top k features which is a logarithmic function of the total number of features. Table II reveals the ranking of each feature from 1 to 82 for each of the 7 projects for the two feature ranking techniques. Similarly, we compute the rank of each feature for Gain Ratio (GR), RELEIF and Symmetric Uncertainty (SU) ranking techniques. Table II shows that the rank of the feature UniqueDerefSet is 1 for Projects 1, 2 and 4 when OneR feature ranking technique is applied. We observe that there are some feature which are ranked highly across projects and across feature selection techniques while there are some features which are ranked higher for a particular project or ranking technique. Table II shows that different feature selection techniques produces different output. An interesting result is that few metrics are highly ranked for some projects while the same metrics is ranked low for other projects within the context of a particular feature selection technique. For example, CountDeclMethod and CountDeclMethodAll are ranked highly for Projects 1, 2 and 4 but low for Projects 5, 6 and 7. We observe that there can be a wide variance in-terms of the ranking for the same metrics across different projects. For example, AltAvgLineBlank metric is ranked as 4 for Project 5 and 82 for Project 1 and Project 3.

TABLE III: Experimental Results using Accuracy, F-Measure and AUC for Multiple Classifiers

SMOTE																		
	ALL			OneR			IG			GR			RF			SU		
Accuracy																		
	ELM-LIN	**ELM-PLY**	**ELM-RBF**	**ELM-LIN**	**ELM-PLY**	**ELM-RBF**	**ELM-LIN**	**ELM-PLY**	**ELM-RBF**	**ELM-LIN**	**ELM-PLY**	**ELM-RBF**	**ELM-LIN**	**ELM-PLY**	**ELM-RBF**	**ELM-LIN**	**ELM-PLY**	**ELM-RBF**
Linux driver net	98.69	86.93	99.56	99.56	99.13	98.04	97.6	98.69	98.69	95.21	83.44	99.56	99.35	93.03	99.56	95.86	86.49	99.56
Linux driver scsi	98.45	89.12	99.48	98.96	99.48	99.48	99.48	99.48	99.48	99.48	99.48	98.96	99.48	99.48	99.48	99.48	99.48	99.48
Linux ext3	83.33	33.33	83.33	66.67	66.67	83.33	50	83.33	83.33	66.67	33.33	83.33	83.33	33.33	83.33	50	66.67	83.33
Linux ipv4	86.96	60.87	91.3	91.3	91.3	91.3	91.3	91.3	82.61	91.3	91.3	86.96	95.65	86.96	91.3	91.3	91.3	82.61
MySQL innodb	82.5	52.5	92.5	95	37.5	28.75	81.25	61.25	90	77.5	91.25	91.25	63.75	82.5	80	76.25	88.75	86.25
MySQL optimizer	75	62.5	87.5	62.5	62.5	87.5	87.5	87.5	87.5	87.5	87.5	87.5	87.5	50	87.5	75	87.5	87.5
MySQL replication	57.14	57.14	85.71	71.43	85.71	85.71	71.43	100	85.71	57.14	85.71	85.71	85.71	100	85.71	100	57.14	85.71
F-Measure																		
Linux driver net	0.99	0.93	1	1	1	0.99	0.99	0.99	0.99	0.98	0.91	1	1	0.96	1	0.98	0.93	1
Linux driver scsi	0.99	0.94	1	0.99	1	1	1	1	1	1	1	0.99	1	1	1	1	1	1
Linux ext3	0.89	0.5	0.91	0.75	0.75	0.91	0.67	0.91	0.91	0.75	0.5	0.91	0.89	0.5	0.91	0.67	0.75	0.91
Linux ipv4	0.93	0.73	0.95	0.95	0.95	0.95	0.95	0.95	0.9	0.95	0.95	0.93	0.98	0.93	0.95	0.95	0.95	0.9
MySQL innodb	0.9	0.67	0.96	0.97	0.52	0.42	0.89	0.74	0.95	0.87	0.95	0.95	0.77	0.9	0.89	0.85	0.94	0.93
MySQL optimizer	0.86	0.73	0.93	0.77	0.77	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.92	0.67	0.93	0.86	0.93	0.93
MySQL replication	0.67	0.67	0.92	0.8	0.92	0.92	0.83	1	0.92	0.67	0.92	0.92	0.92	1	0.92	1	0.73	0.92
Linux driver net	0.5	0.44	0.5	0.5	0.5	0.49	0.49	0.5	0.5	0.48	0.67	0.5	0.5	0.72	0.5	0.48	0.43	0.5
AUC																		
Linux driver scsi	0.49	0.45	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Linux ext3	0.9	0.2	0.5	0.8	0.8	0.5	0.3	0.5	0.5	0.8	0.2	0.5	0.9	0.2	0.5	0.3	0.8	0.5
Linux ipv4	0.48	0.79	0.5	0.5	0.5	0.5	0.5	0.5	0.45	0.5	0.5	0.48	0.75	0.48	0.5	0.5	0.5	0.45
MySQL innodb	0.68	0.51	0.5	0.9	0.43	0.31	0.75	0.71	0.49	0.57	0.65	0.57	0.57	0.68	0.51	0.8	0.79	0.47
MySQL optimizer	0.43	0.79	0.5	0.36	0.36	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.93	0.29	0.5	0.43	0.5	0.5
MySQL replication	0.75	0.75	0.5	0.83	0.5	0.5	0.42	1	0.5	0.75	0.5	0.5	0.5	1	0.5	1	0.33	0.5
Without SMOTE																		
	ALL			OneR			IG			GR			RF			SU		
Accuracy																		
	ELM-LIN	**ELM-PLY**	**ELM-RBF**	**ELM-LIN**	**ELM-PLY**	**ELM-RBF**	**ELM-LIN**	**ELM-PLY**	**ELM-RBF**	**ELM-LIN**	**ELM-PLY**	**ELM-RBF**	**ELM-LIN**	**ELM-PLY**	**ELM-RBF**	**ELM-LIN**	**ELM-PLY**	**ELM-RBF**
Linux driver net	99.35	77.78	99.56	99.56	99.56	98.69	97.17	98.91	99.56	98.47	73.64	99.56	99.56	95.86	99.56	98.04	7.41	99.56
Linux driver scsi	98.96	77.2	99.48	98.96	99.48	98.96	99.48	99.48	99.48	99.48	99.48	99.48	99.48	99.48	98.45	99.48	99.48	99.48
Linux ext3	83.33	50	83.33	16.67	66.67	83.33	50	100	83.33	83.33	50	83.33	50	100	83.33	50	66.67	83.33
Linux ipv4	95.65	47.83	91.3	91.3	91.3	91.3	91.3	91.3	91.3	91.3	91.3	86.96	91.3	91.3	91.3	91.3	91.3	91.3
MySQL innodb	88.75	75	92.5	92.5	80	88.75	90	92.5	90	91.25	92.5	92.5	68.75	90	88.75	90	95	90
MySQL optimizer	62.5	75	87.5	62.5	62.5	87.5	87.5	87.5	87.5	87.5	75	87.5	87.5	75	87.5	75	75	87.5
MySQL replication	85.71	57.14	85.71	85.71	85.71	85.71	71.43	85.71	85.71	14.29	100	85.71	42.86	57.14	85.71	85.71	28.57	85.71
Linux driver net	1	0.87	1	1	1	0.99	0.99	0.99	1	0.99	0.85	1	1	0.98	1	0.99	0.13	1
F-Measure																		
Linux driver scsi	0.99	0.87	1	0.99	1	0.99	1	1	1	1	1	1	1	1	0.99	1	1	1
Linux ext3	0.89	0.67	0.91	0.29	0.75	0.91	0.57	1	0.91	0.91	0.67	0.91	0.67	1	0.91	0.67	0.8	0.91
Linux ipv4	0.98	0.6	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.93	0.95	0.95	0.95	0.95	0.95	0.95
MySQL innodb	0.94	0.85	0.96	0.96	0.88	0.94	0.95	0.96	0.95	0.95	0.96	0.96	0.81	0.95	0.94	0.95	0.97	0.95
MySQL optimizer	0.73	0.83	0.93	0.73	0.77	0.93	0.93	0.92	0.93	0.93	0.86	0.93	0.93	0.86	0.93	0.86	0.86	0.93
MySQL replication	0.91	0.73	0.92	0.91	0.91	0.92	0.83	0.91	0.92	0.25	1	0.92	0.6	0.73	0.92	0.91	0.44	0.92
AUC																		
Linux driver net	0.5	0.64	0.5	0.5	0.5	0.5	0.49	0.5	0.5	0.49	0.87	0.5	0.5	0.73	0.5	0.49	0.54	0.5
Linux driver scsi	0.5	0.39	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.49	0.5	0.5	0.5
Linux ext3	0.9	0.3	0.5	0.1	0.8	0.5	0.7	1	0.5	0.5	0.3	0.5	0.3	1	0.5	0.3	0.4	0.5
Linux ipv4	0.75	0.71	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.48	0.5	0.5	0.5	0.5	0.5	0.5
MySQL innodb	0.71	0.64	0.5	0.58	0.82	0.56	0.49	0.58	0.49	0.57	0.5	0.58	0.45	0.49	0.56	0.56	0.74	0.49
MySQL optimizer	0.79	0.86	0.5	0.79	0.36	0.5	0.5	0.93	0.5	0.5	0.43	0.5	0.5	0.43	0.5	0.43	0.43	0.5
MySQL replication	0.92	0.33	0.5	0.92	0.92	0.5	0.42	0.92	0.5	0.08	1	0.5	0.25	0.33	0.5	0.92	0.17	0.5

TABLE IV: Descriptive Statistics of the Relative Performance of Various Techniques in-terms of Accuracy, F-Measure and AUC

Accuracy														
	Without SMOTE							**With SMOTE**						
	Min	**Max**	**Mean**	**Median**	**Std Dev**	**Q1**	**Q3**	**Min**	**Max**	**Mean**	**Median**	**Std Dev**	**Q1**	**Q3**
Accuracy														
ELM-LIN	14.29	99.56	81.97	90.00	21.61	75.00	98.04	50.00	100.00	83.89	87.50	14.92	75.00	97.60
ELM-PLY	7.41	100.00	79.87	86.61	21.02	73.64	95.86	33.33	100.00	78.59	86.95	20.62	62.50	91.30
ELM-RBF	83.33	99.56	90.88	89.38	5.97	85.71	98.69	28.75	99.56	88.70	87.50	11.45	85.71	98.69
ALL	47.83	99.56	81.60	85.71	15.84	75.00	93.29	33.33	99.56	79.23	85.71	18.30	62.09	91.60
OneR	16.67	99.56	84.13	88.75	19.28	82.50	98.76	28.75	99.56	81.04	87.50	20.47	66.67	98.27
IG	50.00	100.00	89.48	91.30	11.49	87.05	99.05	50.00	100.00	87.02	87.50	13.08	83.15	98.69
GR	14.29	100.00	84.88	91.25	19.94	83.33	98.72	33.33	99.56	84.77	87.50	15.74	83.41	92.28
RF	42.86	100.00	84.90	90.00	16.86	81.25	98.71	33.33	100.00	85.09	87.50	17.16	83.12	99.38
SU	7.41	99.56	80.47	90.00	24.32	75.00	95.76	50.00	100.00	85.22	87.50	13.81	81.02	96.77
F-Measure														
	Without SMOTE							**With SMOTE**						
	Min	**Max**	**Mean**	**Median**	**Std Dev**	**Q1**	**Q3**	**Min**	**Max**	**Mean**	**Median**	**Std Dev**	**Q1**	**Q3**
ELM-LIN	0.25	1.00	0.88	0.95	0.18	0.86	0.99	0.67	1.00	0.90	0.93	0.10	0.85	0.99
ELM-PLY	0.13	1.00	0.86	0.92	0.17	0.83	0.98	0.50	1.00	0.86	0.93	0.16	0.74	0.95
ELM-RBF	0.91	1.00	0.95	0.95	0.03	0.92	0.99	0.42	1.00	0.93	0.93	0.09	0.92	0.99
ALL	0.60	1.00	0.88	0.91	0.11	0.85	0.97	0.50	1.00	0.86	0.92	0.14	0.73	0.95
OneR	0.29	1.00	0.89	0.94	0.16	0.90	0.99	0.42	1.00	0.87	0.93	0.16	0.77	0.99
IG	0.57	1.00	0.93	0.95	0.09	0.92	0.99	0.67	1.00	0.92	0.93	0.09	0.91	0.99
GR	0.25	1.00	0.90	0.95	0.17	0.91	0.99	0.50	1.00	0.90	0.93	0.12	0.91	0.96
RF	0.60	1.00	0.91	0.95	0.11	0.90	0.99	0.50	1.00	0.91	0.93	0.12	0.90	1.00
SU	0.13	1.00	0.86	0.95	0.21	0.86	0.98	0.67	1.00	0.91	0.93	0.09	0.89	0.99
AUC														
	Without SMOTE							**With SMOTE**						
	Min	**Max**	**Mean**	**Median**	**Std Dev**	**Q1**	**Q3**	**Min**	**Max**	**Mean**	**Median**	**Std Dev**	**Q1**	**Q3**
ELM-LIN	0.08	0.92	0.53	0.50	0.19	0.49	0.57	0.30	1.00	0.59	0.50	0.18	0.49	0.75
ELM-PLY	0.17	1.00	0.58	0.50	0.22	0.43	0.74	0.20	1.00	0.55	0.50	0.19	0.48	0.68
ELM-RBF	0.48	0.58	0.50	0.50	0.02	0.50	0.50	0.31	0.57	0.49	0.50	0.03	0.50	0.50
ALL	0.30	0.92	0.59	0.50	0.18	0.50	0.72	0.20	0.90	0.56	0.50	0.16	0.49	0.70
OneR	0.10	0.92	0.56	0.50	0.19	0.50	0.63	0.31	0.90	0.54	0.50	0.16	0.50	0.50
IG	0.42	1.00	0.57	0.50	0.17	0.50	0.52	0.30	1.00	0.53	0.50	0.14	0.50	0.50
GR	0.08	1.00	0.51	0.50	0.18	0.50	0.50	0.20	0.80	0.53	0.50	0.12	0.50	0.57
RF	0.25	1.00	0.50	0.50	0.15	0.48	0.50	0.20	1.00	0.57	0.50	0.20	0.50	0.69
SU	0.17	0.92	0.50	0.50	0.14	0.48	0.50	0.30	1.00	0.54	0.50	0.17	0.47	0.50

Relative Comparison of Techniques: Table III displays a complete analysis and covers all the results in-terms of the three performance metrics. Table III displays results for all the combinations of the feature selection technique, with and without-SMOTE and the three ELM kernel functions. From Table III, we infer that the performance of the prediction model depends on the dataset, feature selection technique and the kernel function. While some approaches performs better than other approaches in certain cases, we do not observe any one particular technique dominating all other strategies or any one particular strategy. Table IV displays the descriptive statistics of the relative performance of various techniques in-terms of accuracy, f-measure and AUC. There are 7 projects, 5 different techniques for feature selection, 1 strategy for imbalance learning (resulting in 2 variations) and 3 ELM kernel methods. Hence we generate $7 * 2 * 3 = 42$ data points to compute the mean value of any feature selection technique, impact of the imbalance learning strategy or ELM kernel method.

Table IV reveals that the mean AUC value for ELM-LIN classifier with SMOTE is 0.59 whereas the mean AUC value for ELM-LIN classifier without SMOTE is 0.53. This result indicate that the application of SMOTE technique for linear kernel improves the performance. However, the result for ELM-PLY and ELM-RBF is different than the result from ELM-LIN. The mean AUC value for ELM-PLY and ELM-RBF is higher for without-SMOTE than the corresponding mean value for with-SMOTE. Table V displays the descriptive statistics of the relative performance of SMOTE and without SMOTE in-terms of AUC. The descriptive statistics is obtained by obtaining AUV values from $7 * 5 * 3 = 105$ classifiers. Table V reveals that the with-SMOTE technique performs equal to without-SMOTE in-terms of the median AUC value but outperforms in-terms of the mean AUC value.

TABLE V: Descriptive Statistics of the Relative Performance of SMOTE and Without SMOTE in-terms of AUC

AUC							
	Min	**Max**	**Mean**	**Median**	**Std Dev**	**Q1**	**Q3**
Without Smote	0.080	1.000	0.541	0.500	0.169	0.500	0.540
With Somte	0.200	1.000	0.546	0.500	0.159	0.500	0.510

Null Hypothesis Statistical Significance Testing: We apply the nonparametric Wilcoxon signed-rank test with a Bonfer-

TABLE VI: t-test

(a) t-test: Among Different Kernels

p-value			
	ELM-LIN	**ELM-PLY**	**ELM-RBF**
ELM-LIN	NaN	0.63	0.00
ELM-PLY	0.63	NaN	0.00
ELM-RBF	0.00	0.00	NaN

(b) t-test: Among Different Sets of Metrics

p-value						
	ALL	**OneR**	**IG**	**GR**	**RF**	**SU**
ALL	NaN	0.55	0.52	0.15	0.36	0.11
OneR	0.55	NaN	1.00	0.43	0.72	0.25
IG	0.52	1.00	NaN	0.40	0.67	0.39
GR	0.15	0.43	0.40	NaN	0.65	0.89
RF	0.36	0.72	0.67	0.65	NaN	0.64
SU	0.11	0.25	0.39	0.89	0.64	NaN

roni correction to compare whether the two populations are different. It is a type of paired t-test used to check if the null hypothesis is true or false [17]. Table VI shows the output of the hypothesis testing consisting of a comparison between five feature selection techniques and three different ELM kernels based on AUC performance metrics. In Table VI, a value below 0.05 means that H_0 is rejected and a value above 0.05 means that H_0 is accepted and there is no statistically significant difference between the two techniques under comparison. We use a standard cut-off of 0.05 (the null hypothesis is rejected when $p < 0.05$ and not rejected when $p > 0.05$). Table VI reveals several values below 0.05 which shows that there is a statistically significant difference between the performances of the various techniques. We observe that there is a statistically significant difference in the performance of ELM with linear and RBF kernel as well as polynomial and RBF kernel. However, there is no statistically significant difference between ELM with linear and polynomial kernel.

V. Answer to Research Questions

Answer RQ 1: There is no statistically significant difference between the 10 pair-wise comparisons (5 different techniques excluding the all metrics). All the five feature selection technique performs equally well in-terms of the statistical significance test.

Answer RQ 2: There is a statistically significant difference without-SMOTE and with-SMOTE. The mean AUC value with-SMOTE is hgiher than the mean AUC value without-SMOTE

Answer RQ 3: There is a statistically significant difference between the performance of 3 different ELM kernel functions according to the Wilcoxon signed-rank test. ELM with linear kernel function outperforms ELM with polynomial and RBF kernel.

VI. Conclusion

Our main conclusions are that static source code metrics can be used as predictors for aging related bugs. We conclude that while there are several source code metrics which can be extracted from the software system, not all are equally important as several metrics are redundant and irrelevant. Our results reveal that some of the feature ranking techniques for dimensionality reduction improves the performance of the predictive model in comparison to using all metrics but there is no statistically significant difference in their performance. Imbalanced and highly skewed data is one of the major challenges in learning a predictive model and classifying unseen instanced. We apply SMOTE based strategy to counter the effect of imbalanced data distribution and our empirical result shows that SMOTE improves the performance for ELM linear kernel but not polynomial or RBF kernel. We observe statistically significant difference between the performance of 3 different ELM kernel functions according to the Wilcoxon signed-rank test.

References

[1] D. Cotroneo, R. Natella, and R. Pietrantuono, "Is software aging related to software metrics?" in *Software Aging and Rejuvenation (WoSAR), 2010 IEEE Second International Workshop on*. IEEE, 2010, pp. 1–6.

[2] D. Cotroneo, R. Natella, R. Pietrantuono, and S. Russo, "Software aging analysis of the linux operating system," in *Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium on*. IEEE, 2010, pp. 71–80.

[3] 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.

[4] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[5] M. A. Hall, "Correlation-based feature selection for machine learning," 1999.

[6] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Transactions on knowledge and data engineering*, vol. 17, no. 4, pp. 491–502, 2005.

[7] L. Kumar and A. Sureka, "Using structured text source code metrics and artificial neural networks to predict change proneness at code tab and program organization level." in *ISEC*, 2017, pp. 172–180.

[8] L. Kumar, S. K. Rath, and A. Sureka, "Empirical analysis on effectiveness of source code metrics for predicting change-proneness." in *ISEC*, 2017, pp. 4–14.

[9] "The promise repository of empirical software engineering data," 2015.

[10] R. Natella, "Aging-related bugs and software complexity metrics Aging-related bugs and software complexity metrics," May 2017. [Online]. Available: https://doi.org/10.5281/zenodo.581659

[11] J. Sandberg and M. Alvesson, "Ways of constructing research questions: gap-spotting or problematization?" *Organization*, vol. 18, no. 1, pp. 23–44, 2011.

[12] N. V. Chawla, "Data mining for imbalanced datasets: An overview," in *Data mining and knowledge discovery handbook*. Springer, 2005, pp. 853–867.

[13] T. R. Hoens and N. V. Chawla, "Imbalanced datasets: from sampling to classifiers," *Imbalanced Learning: Foundations, Algorithms, and Applications*, pp. 43–59, 2013.

[14] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[15] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.

[16] M. Sokolova and G. Lapalme, "Performance measures in classification of human communications," *Advances in Artificial Intelligence*, pp. 159–170, 2007.

[17] R. Woolson, "Wilcoxon signed-rank test," *Wiley encyclopedia of clinical trials*, 2008.