Feature Selection Techniques to Counter Class Imbalance Problem for Aging Related Bug Prediction

Lov Kumar[1] Ashish Sureka[2]

[1]Thapar University, India (lovkumar505@gmail.com)

[2]Ashoka University, India (ashish.sureka@ashoka.edu.in)

ISEC 2018 [9-11 February 2018, Hyderabad, India]

Table of Contents

1. Research Motivation and Aim
   - Objectives and Context Setting
   - Research Questions
2. Literature Survey
3. Research Contributions - 1
4. Research Framework and Solution Approach
5. Experimental Dataset
6. Experimental Results
7. Answer to Research Questions
8. Conclusion

Table of Contents

1. Research Motivation and Aim
   - Objectives and Context Setting
   - Research Questions
2. Literature Survey
3. Research Contributions - 1
4. Research Framework and Solution Approach
5. Experimental Dataset
6. Experimental Results
7. Answer to Research Questions
8. Conclusion

Aging-Related Bugs (ARBs)

Error-conditions Associated with Software Aging

Aging-Related Bugs (ARBs) are defined as those bugs which are caused by error-conditions associated with software aging.

System crash or hang due to gradual depletion of resources (such as memory leak in a web server) [7][9][8]

Happens when the execution time or period of the system is long

Error conditions like memory leakage or unreleased files/locks that accumulate over period of time before failure is triggered [7][9][8].

Problem Encountered by Software Developers

Hard to Uncover Aging Related Bugs

Non-trivial to replicate such bugs

Our Motivation: Early prediction of such bugs and identification of files or classes in a software system

Recent research shows that static source code metrics can be used as predictors for aging related bugs within a machine learning framework [7][9][8].

Technical Challenges

Imbalanced Data

Class of interest is in minority which poses technical challenges in building an effective classifier

Class of interest (aging related bugs) has only 1% instances in comparison to 99% instances of the majority class - the dataset is considered to be highly imbalanced

Technical Challenges

High-Dimensional Data

High dimensional feature space consisting of irrelevant and redundant features decreases the performance of the classifiers [11] [12] [19]

Presence of large number of features (in our case, static source code features) poses intrinsic challenges to machine learning classification algorithms

Research Motivation

Prediction Problem and Automation

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?

Aim: building tool support for software developers for identifying aging related fault-prone classes

Table of Contents

1. Research Motivation and Aim
   - Objectives and Context Setting
   - Research Questions
2. Literature Survey
3. Research Contributions - 1
4. Research Framework and Solution Approach
5. Experimental Dataset
6. Experimental Results
7. Answer to Research Questions
8. Conclusion

Research Questions - RQ1 and RQ2

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: Does removing a significant percentage of features improves the predictive performance substantially or does the model performance remains similar even after applying feature ranking or selection techniques?

Research Questions - RQ3 and RQ4

RQ 3: Is there a statistically significant difference in the prediction performance of five different techniques applied to address the class imbalance problem? Which of the five technique(s) yields the best result?

RQ 4: What is the relative performance of the five different types of learning algorithms and classifiers in-terms of the AUC, accuracy and f-measure metrics? Is there a statistically significant difference in the prediction performance of five different classifiers?

Related Work - 1

Cotroneo et al. [9]

Cotroneo et al. presents analysis of software gaining phenomenon at the Operating System (OS) level and investigate the software aging sources inside the Linux OS kernel [9]

Qin et al. [22]

Qin et al. investigate if aging related bugs can be identified using cross-project prediction approaches [22].

Related Work - 2

Cotroneo et al. [7]

Cotroneo et al. conduct an empirical study to investigate the relationship between static features of the software and aging [7].

Wang et al. [27]

Wang et al. study the issue of if and how class imbalance learning methods can benefit software defect prediction with the aim of finding better solutions [27]

Research Contributions

The study presented in this paper is the *first study* on the application of five different feature selection techniques, five different strategies to counter the effect of imbalance data and five different machine learning algorithms for predicting aging related bugs in seven sub-systems of two large open-source software projects using 82 source code metrics as predictors.

Research Contributions - 2

We conduct a series of experiments to measure the performance of various techniques using metrics such as accuracy, f-measure and AUC and present results showing the effectiveness of various approaches.

We frame *four research questions* and conduct statistical significance test to answer the stated research questions on identifying the best feature selection technique, imbalance learning strategy and classification algorithm.

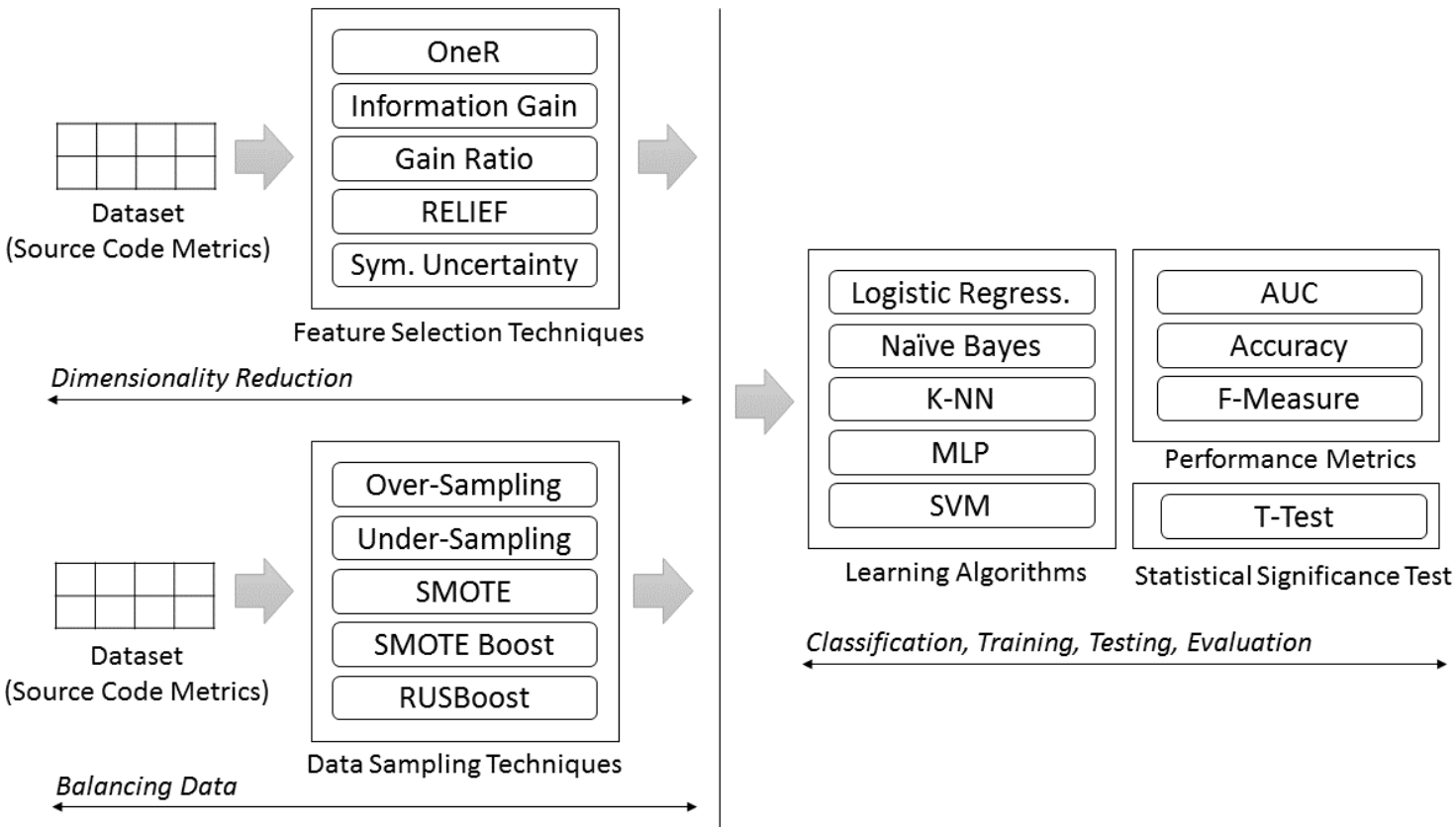
Solution Approach

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.

Multi-Step Process



Feature Ranking and Selection

Reduce dimensionality of input

Reduce the number of redundant and irrelevant features and noise in the data

Higher ranked features have better prediction capability than the features which are lower ranked

We apply five different feature selection approaches: OneR, Information Gain (IG), Gain Ratio (GR), RELEIF (RF) and Symmetric Uncertainty (SU).

Learning from Imbalanced Dataset

Issue of class imbalance

Two approaches: data-level and classification algorithm level.

We apply five different techniques to address the class imbalance problem: Random Under-sampling (USAM), Random Oversampling (OSAM), SMOTE, SMOTEBoost and RUSBoost.

The data level approach consists of oversampling the rare or minority class (the class which is of interest) or under-sampling the majority class.

Learning Algorithms

Classifiers and Predictive Models

We apply five different types of learning algorithms

We apply both classical algorithms as well as relatively modern and new statistical algorithms

There is a wide variance in the performance of the various learning algorithms on the same dataset.

Performance Evaluation

Measuring effectiveness of a classifier

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

We use AUC as AUC has an attractive property of being insensitive to changes in the class distribution [26]

Experimental Dataset

We use a publicly available dataset available at tera-PROMISE Repository[a] for our experiments.

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

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

Experimental Data Set - Class Distribution

ID	Name	Records	Majority	Minority	% Majority	% Minority
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

Experimental Dataset

The dataset by Roberto et al. [20] has been used in the past for building aging-related defect prediction models

The dataset contains several types of metrics (82 metrics - such as program size related metrics, McCabe's Cyclomatic complexity, Halstead metrics and aging-related metrics [20].

Feature Ranking Results - 1

We select top k features which is a logarithmic function of the total number of features.

Rank of the feature UniqueDerefSet is 1 for Projects 1, 2 and 4 when OneR feature ranking technique is applied

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

OneR Feature Ranking Results - 1

	AUC					Accuracy				
Undersampling										
	LOGR	**NBC**	**MLP**	**KNN**	**SVM**	**LOGR**	**NBC**	**MLP**	**KNN**	**SVM**
Linux driver net	0.81	0.82	0.75	0.78	0.54	92.03	93.33	90.81	85.58	98.56
Linux driver scsi	0.66	0.62	0.51	0.57	0.48	72.33	83.63	81.87	72.85	94.82
Linux ext3	0.54	0.6	0.48	0.62	0.5	63.33	73.33	66.67	63.33	70
Linux ipv4	0.34	0.39	0.4	0.38	0.45	66.09	77.39	78.26	73.91	88.7
MySQL innodb	0.68	0.73	0.75	0.67	0.69	80.25	83.25	79.5	76.5	88.5
MySQL optimizer	0.8	0.8	0.73	0.83	0.86	80	80	82.5	85	90
MySQL replication	0.67	0.65	0.67	0.77	0.67	85.71	82.86	85.71	88.57	85.71
Oversampling										
	LOGR	**NBC**	**MLP**	**KNN**	**SVM**	**LOGR**	**NBC**	**MLP**	**KNN**	**SVM**
Linux driver net	0.86	0.72	0.65	0.55	0.85	92.33	93.51	90.68	99.04	89.24
Linux driver scsi	0.66	0.53	0.47	0.5	0.47	91.71	84.87	94.2	98.76	93.16
Linux ext3	0.6	0.82	0.68	0.7	0.72	73.33	83.33	73.33	76.67	80
Linux ipv4	0.42	0.39	0.43	0.48	0.41	81.74	77.39	85.22	94.78	80
MySQL innodb	0.82	0.8	0.74	0.48	0.82	91.75	89	88.75	86.5	91.5
MySQL optimizer	0.8	0.79	0.53	0.66	0.86	95	77.5	77.5	85	90
MySQL replication	0.67	0.87	0.87	0.87	0.87	85.71	91.43	91.43	91.43	91.43

OneR Feature Ranking Results - 2

	SMOTE									
	LOGR	NBC	MLP	KNN	SVM	LOGR	NBC	MLP	KNN	SVM
Linux driver net	0.5	0.74	0.5	0.49	0.5	99.56	97.17	99.56	98.04	99.56
Linux driver scsi	0.5	0.5	0.5	0.99	0.5	99.48	99.48	99.48	98.45	99.48
Linux ext3	0.7	0.3	0.7	0.8	0.3	50	50	50	66.67	50
Linux ipv4	0.5	0.5	0.5	0.5	0.5	91.3	91.3	91.3	91.3	91.3
MySQL innodb	0.78	0.76	0.54	0.61	0.7	87.5	83.75	85	83.75	87.5
MySQL optimizer	0.5	0.5	0.5	0.5	0.5	87.5	87.5	87.5	87.5	87.5
MySQL replication	0.92	0.92	0.92	0.42	0.92	85.71	85.71	85.71	71.43	85.71
	SMOTEBoost									
	LOGR	NBC	MLP	KNN	SVM	LOGR	NBC	MLP	KNN	SVM
Linux driver net	0.95	0.5	0.49	0.48	0.95	90.63	98.91	97.82	96.51	90.85
Linux driver scsi	0.93	0.5	0.5	0.5	0.95	86.01	99.48	99.48	99.48	90.67
Linux ext3	0.3	0.5	0.9	0.9	0.8	50	83.33	83.33	83.33	66.67
Linux ipv4	0.45	0.5	0.5	0.5	0.68	82.61	91.3	91.3	91.3	82.61
MySQL innodb	0.7	0.54	0.44	0.54	0.63	86.25	86.25	81.25	85	87.5
MySQL optimizer	0.93	0.5	0.93	0.93	1	87.5	87.5	87.5	87.5	100
MySQL replication	0.5	1	0.5	1	1	85.71	100	85.71	100	100

Feature Ranking Results - 2

Different feature selection techniques produces different output

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

CountDeclMethod and CountDeclMethodAll are ranked highly for Projects 1, 2 and 4 but low for Projects 5, 6 and 7.

Relative Comparison of Techniques - 1

There are 7 projects, 5 different techniques for feature selection, 5 strategies for imbalance learning and 5 learning algorithms.

Mean AUC value of USAM is best in comparison to the other imbalance learning strategies.

According to the f-measure value, SMOTEBoost performs best followed by RUSBoost.

Relative Comparison of Techniques - 2

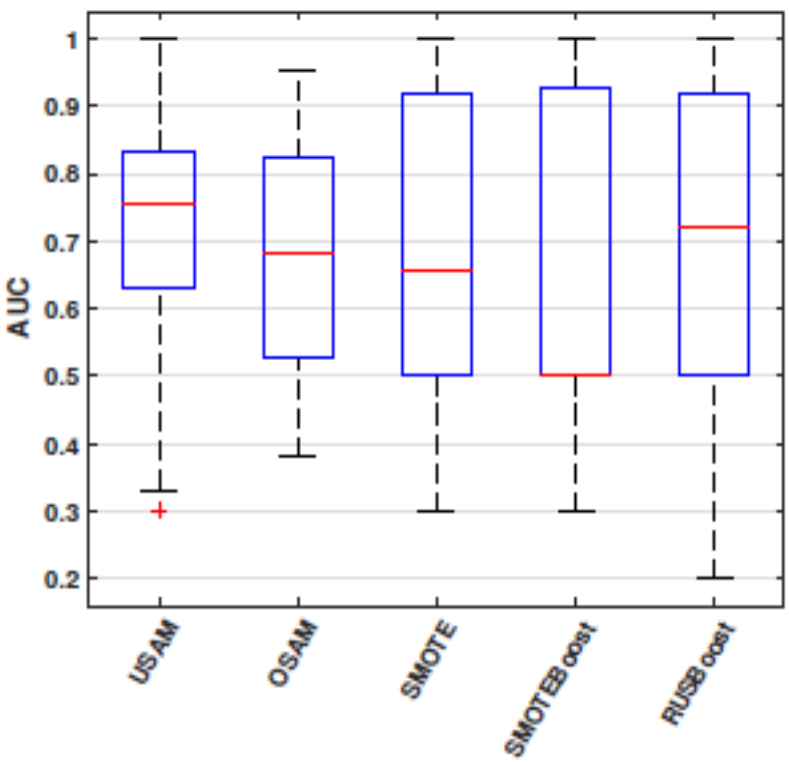
SVM has a mean AUC value of 0.71. The mean AUC value of NBC is 0.69 and the mean AUC value of LOGR, MLP and KNN is 0.68 each.

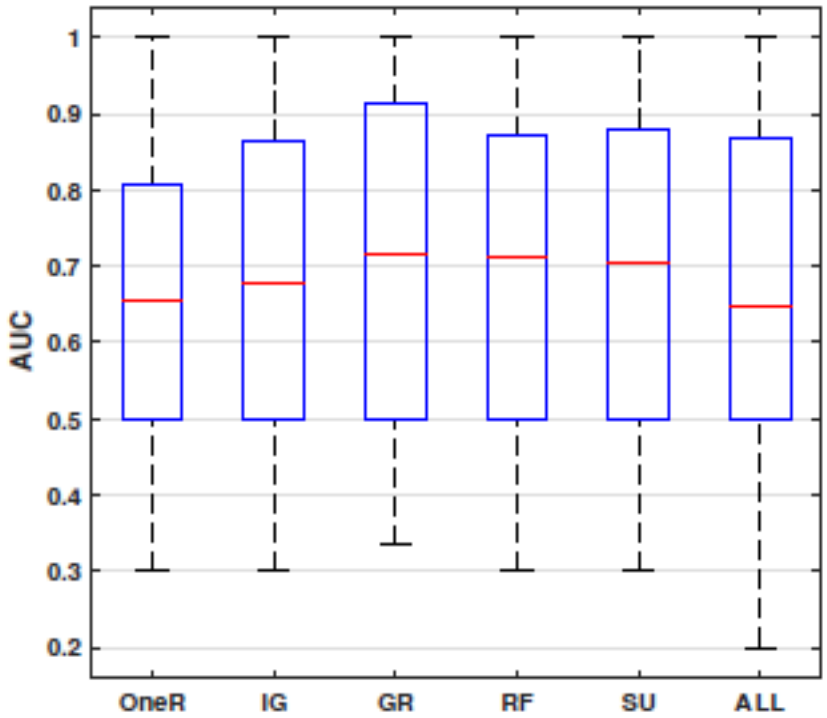
SVM performance best followed by NBC.

AGR and RF performs equally well in-terms of the mean AUC value.

In terms of the f-measure value, IG, GR, RF and SU performs equally well having a mean value of 0.93.

Boxplot - 1


AUC
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
USAM
OSAM
SMOTE
SMOTEBoost
RUSBoost


AUC
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
OneR
IG
GR
RF
SU
ALL

Descriptive Statistics - AUC

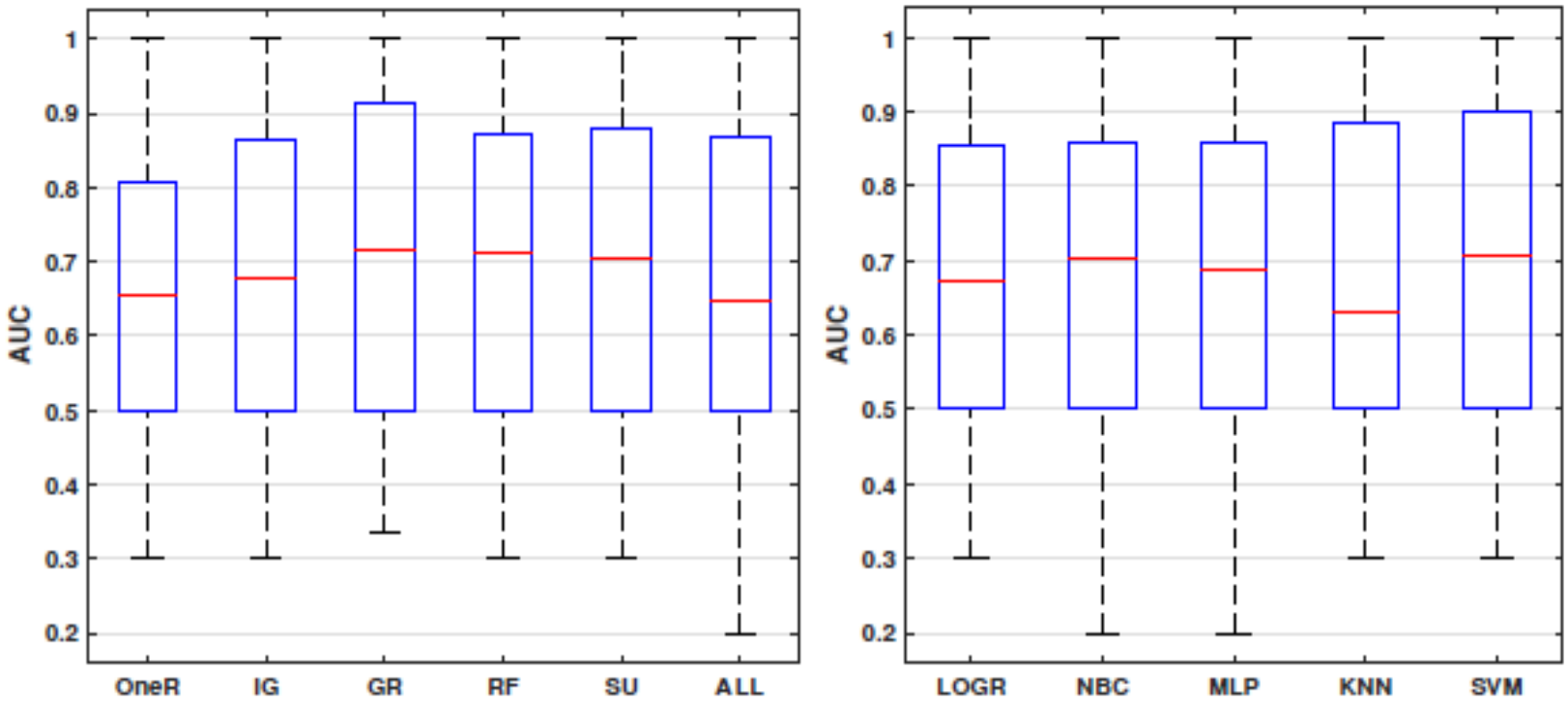
AUC							
	Min	Max	Mean	Median	Std Dev	Q1	Q3
USAM	0.30	1.00	0.72	0.76	0.16	0.63	0.83
OSAM	0.38	0.95	0.68	0.68	0.16	0.53	0.82
SMOTE	0.30	1.00	0.69	0.66	0.21	0.50	0.92
SMOTEBoost	0.30	1.00	0.66	0.50	0.22	0.50	0.93
RUSBoost	0.20	1.00	0.70	0.72	0.22	0.50	0.92
OneR	0.30	1.00	0.66	0.65	0.19	0.50	0.81
IG	0.30	1.00	0.68	0.68	0.20	0.50	0.87
GR	0.33	1.00	0.71	0.72	0.20	0.50	0.91
RF	0.30	1.00	0.71	0.71	0.19	0.50	0.87
SU	0.30	1.00	0.70	0.71	0.20	0.50	0.88
ALL	0.20	1.00	0.68	0.65	0.21	0.50	0.87
LOGR	0.30	1.00	0.68	0.67	0.20	0.50	0.86
NBC	0.20	1.00	0.69	0.70	0.19	0.50	0.86
MLP	0.20	1.00	0.68	0.69	0.19	0.50	0.86
KNN	0.30	1.00	0.68	0.63	0.20	0.50	0.89
SVM	0.30	1.00	0.71	0.71	0.20	0.50	0.90

Distribution Comparison, Visualization [Boxplot] - 1

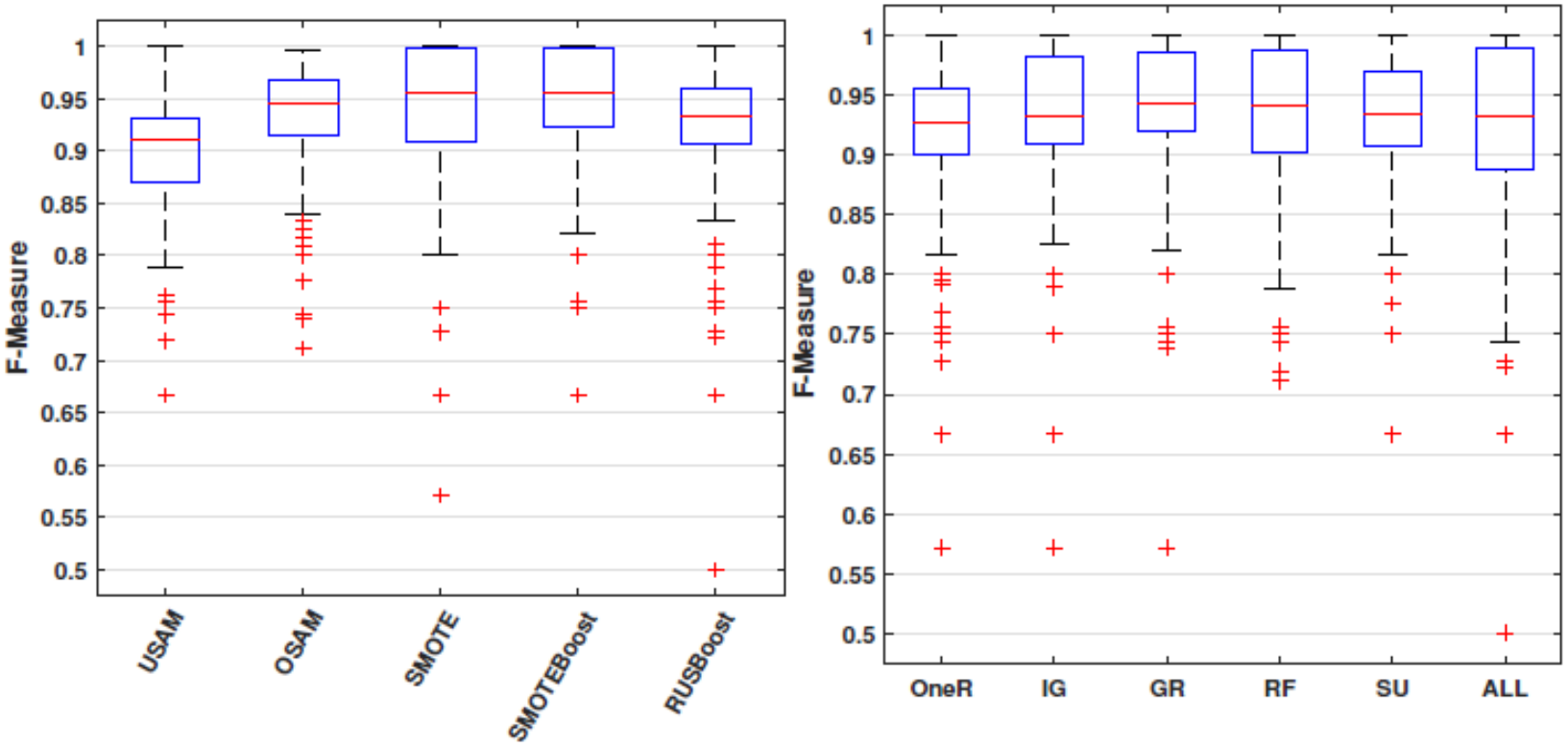
From Boxplot: median AUC value of USAM is higher than the corresponding values for other imbalance learning techniques.

The inter-quartile range (the middle box representing the middle 50% of the values of the variable) range for GR is more than the corresponding values for OneR, IG, RF, SU and ALL.

Boxplot - 2


AUC
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
OneR
IG
GR
RF
SU
ALL
AUC
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
LOGR
NBC
MLP
KNN
SVM

Boxplot - 3


F-Measure
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
USAM
OSAM
SMOTE
SMOTEBoost
RUSBoost
F-Measure
OneR
IG
GR
RF
SU
ALL

Distribution Comparison, Visualization [Boxplot] - 2

The boxplot for the AUC values for NBC and MLP is comparatively tall and hence shows that there is more variation in the AUC value obtained from 125 executions.

The span between the first and third quartile is more for SMOTE and SMOTEBoost than USAM, OSAM and RUSBoost.

the span between the first and third quartile for RF and ALL is more for than the span for OneR, IG, GR and SU.

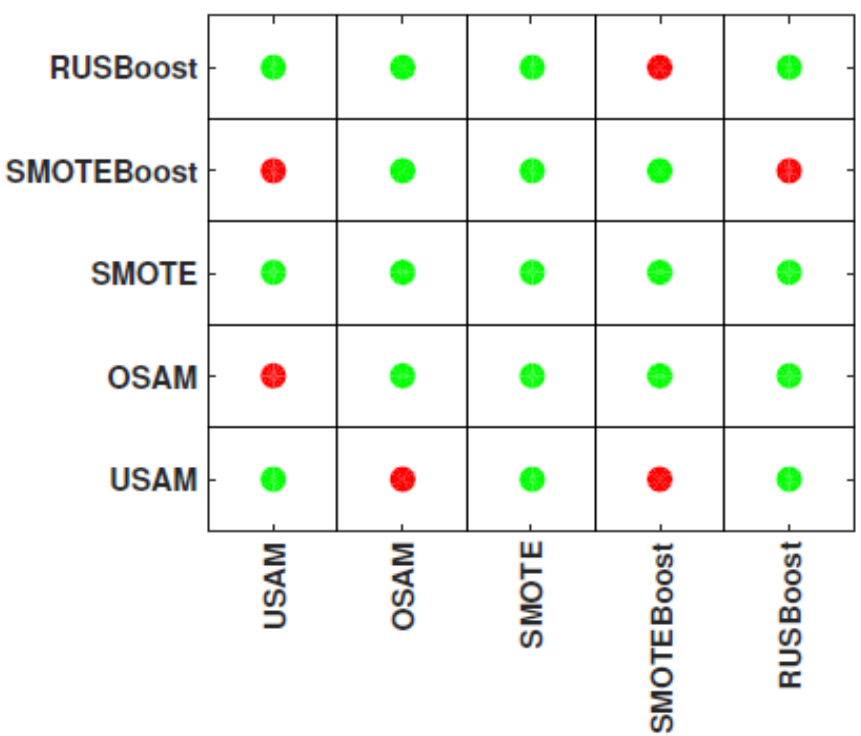
Null Hypothesis Statistical Significance Testing - 1

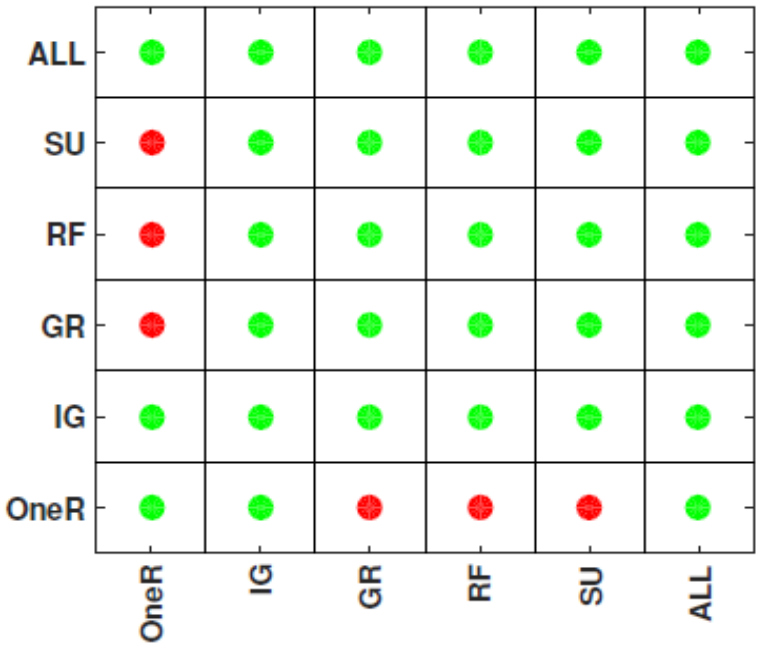
We apply the nonparametric Wilcoxon signed-rank test with a Bonferroni correction to compare whether the two populations are different.

A red dot means that H_0 is rejected and a green dot means that H_0 is accepted

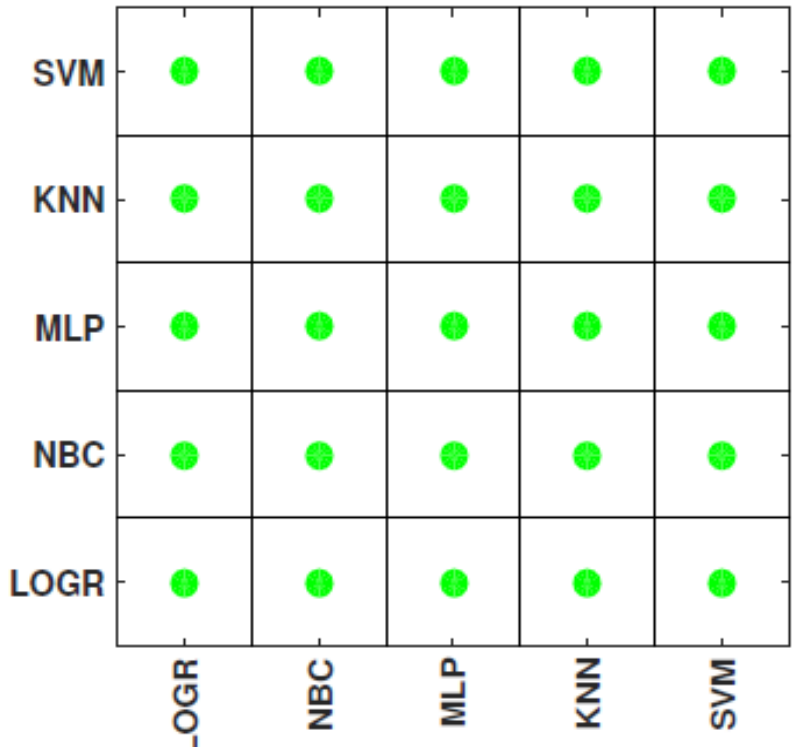
We use a standard cut-off of 0.05 (the null hypothesis is rejected when $p < 0.05$ and not rejected when $p > 0.05$)

Red Dot - H_0 is Rejected





Hypothesis Testing



Null Hypothesis Statistical Significance Testing - 2

Our analysis reveals several red dots which shows that there is a statistically significant difference between the performances of the various techniques.

There is a *statistically significant difference between SMOTEBoost and RUSBoost, OSAM and USAM and USAM and SMOTEBoost*.

We observe that the null hypothesis H_0 is rejected for 3 out of 10 significance tests involving pairwise comparison between the imbalance learning techniques

Answer - Research Question 1

There is a statistically significant difference between 3 out of 10 pair-wise comparisons (5 different techniques excluding the all metrics).

There is a statistically significant difference between OneR and Gain Ratio, OneR and RELIEF and OneR and Symmetric Uncertainty (SU).

In-terms of the f-measure also GR and RF performs best but IG and SU also performs equally well.

Answer - Research Question 2

There is no statistically significant difference between the performance of the all metrics (no filter selection technique) and the 5 different types of feature selection technique according to the Wilcoxon signed-rank test.

However, based on the 125 data points, feature selection technique results in performance improvement in comparison to all metrics.

Answer - Research Question 3

There is a statistical difference between SMOTEBoost and RUSBoost, OSAM and USAM and USAM and SMOTEBoost.

According to the f-measure value, SMOTEBoost performs best followed by RUSBoost. According to AUC, USAM performs best followed by RUSBoost.

Answer - Research Question 4

There is no statistically significant difference between the performance of 5 different classification algorithm according to the Wilcoxon signed-rank test.

SVM performs best in-terms of f-measure and AUC. NBC performs second best in-terms of AUC.

Key Takeaways - 1

Our main conclusions are that static source code metrics can be used as predictors for aging related bugs

Not all source code metric are equally important as several metrics are redundant and irrelevant

Some of the feature ranking techniques for dimensionality reduction improves the performance of the predictive model

Key Takeaways - 2

SMOTEBoost performs best followed by RUSBoost and there is a statistically significant difference between these techniques and other imbalance learning approaches

No statistically significant difference between the performance of 5 different classification algorithm according to the Wilcoxon signed-rank test

References I

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

[2] Gabriella Carrozza, Domenico Cotroneo, Roberto Natella, Roberto Pietrantuono, and Stefano Russo. Analysis and prediction of mandelbugs in an industrial software system. In *Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on*, pages 262–271. IEEE, 2013.

[3] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.

[4] Nitesh Chawla, Aleksandar Lazarevic, Lawrence Hall, and Kevin Bowyer. Smoteboost: Improving prediction of the minority class in boosting. *Knowledge Discovery in Databases: PKDD 2003*, pages 107–119, 2003.

[5] Nitesh V Chawla. Data mining for imbalanced datasets: An overview. In *Data mining and knowledge discovery handbook*, pages 853–867. Springer, 2005.

References II

[6] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[7] Domenico Cotroneo, Roberto Natella, and Roberto Pietrantuono. Is software aging related to software metrics? In *Software Aging and Rejuvenation (WoSAR), 2010 IEEE Second International Workshop on*, pages 1–6. IEEE, 2010.

[8] Domenico Cotroneo, Roberto Natella, and Roberto Pietrantuono. Predicting aging-related bugs using software complexity metrics. *Performance Evaluation*, 70(3):163–178, 2013.

[9] Domenico Cotroneo, Roberto Natella, Roberto Pietrantuono, and Stefano Russo. Software aging analysis of the linux operating system. In *Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium on*, pages 71–80. IEEE, 2010.

[10] Robert Feldt and Ana Magazinius. Validity threats in empirical software engineering research-an initial survey. In *SEKE*, pages 374–379, 2010.

References III

[11] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[12] Mark Andrew Hall. Correlation-based feature selection for machine learning. 1999.

[13] T Ryan Hoens and Nitesh V Chawla. Imbalanced datasets: from sampling to classifiers. *Imbalanced Learning: Foundations, Algorithms, and Applications*, pages 43–59, 2013.

[14] Lov Kumar, Santanu Kumar Rath, and Ashish Sureka. Empirical analysis on effectiveness of source code metrics for predicting change-proneness. In *ISEC*, pages 4–14, 2017.

[15] Lov Kumar and Ashish Sureka. Aging related bug prediction using extreme learning machines. *IEEE India Council International Conferece (INDICON)*, 2017.

[16] Lov Kumar and Ashish Sureka. Using structured text source code metrics and artificial neural networks to predict change proneness at code tab and program organization level. In *ISEC*, pages 172–180, 2017.

References IV

[17] Sangeeta Lal, Neetu Sardana, and Ashish Sureka. Improving logging prediction on imbalanced datasets: A case study on open source java projects. *International Journal of Open Source Software and Processes (IJOSSP)*, 7(2):43–71, 2016.

[18] Tjen-Sien Lim, Wei-Yin Loh, and Yu-Shan Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine learning*, 40(3):203–228, 2000.

[19] Huan Liu and Lei Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on knowledge and data engineering*, 17(4):491–502, 2005.

[20] Roberto Natella. Aging-related bugs and software complexity metrics Aging-related bugs and software complexity metrics, May 2017.

[21] Dewayne E Perry, Adam A Porter, and Lawrence G Votta. Empirical studies of software engineering: a roadmap. In *Proceedings of the conference on The future of Software engineering*, pages 345–355. ACM, 2000.

References V

[22] Fangyun Qin, Zheng Zheng, Chenggang Bai, Yu Qiao, Zhenyu Zhang, and Cheng Chen. Cross-project aging related bug prediction. In *Software Quality, Reliability and Security (QRS), 2015 IEEE International Conference on*, pages 43–48. IEEE, 2015.

[23] Jörgen Sandberg and Mats Alvesson. Ways of constructing research questions: gap-spotting or problematization? *Organization*, 18(1):23–44, 2011.

[24] Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. Rusboost: Improving classification performance when training data is skewed. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008.

[25] Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. Rusboost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 40(1):185–197, 2010.

References VI

[26] Marina Sokolova and Guy Lapalme. Performance measures in classification of human communications. *Advances in Artificial Intelligence*, pages 159–170, 2007.

[27] Shuo Wang and Xin Yao. Using class imbalance learning for software defect prediction. *IEEE Transactions on Reliability*, 62(2):434–443, 2013.

[28] RF Woolson. Wilcoxon signed-rank test. *Wiley encyclopedia of clinical trials*, 2008.