

SOME MACHINE LEARNING ASSISTED SOFTWARE
BUG PREDICTION TECHNIQUES



The thesis submitted in partial fulfilment

for the Award of Degree

DOCTOR OF PHILOSOPHY

by

RAKESH KUMAR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY

(BANARAS HINDU UNIVERSITY)

VARANASI -221005

INDIA

Roll No.: 18071011

Year: 2023

CERTIFICATE

It is certified that the work contained in this thesis entitled "SOME MACHINE LEARNING ASSISTED SOFTWARE BUG PREDICTION TECHNIQUES" submitted by "RAKESH KUMAR" (Roll No.: 18071011) has been carried out under my supervision and that it has not been submitted elsewhere for a degree.

It is further certified that the student has fulfilled all the requirements of Comprehensive Examination, Candidacy and State-of-the-Art (SOTA) for the award of Ph.D. Degree.

Amrita

Signature of Supervisor

Dr. AMRITA CHATURVEDI

Assistant Professor

Department of Computer Science and Engineering

Indian Institute of Technology

(Banaras Hindu University)

Varanasi-221005 , India

पर्यवेक्षक/Supervisor
संगणक विज्ञान एवं अभियंत्रिकी विभाग
Department of Computer Sc. & Engg
भारतीय प्रौद्योगिकी संस्थान
Indian Institute of Technology
(काशी हिन्दू विश्वविद्यालय)
(Banaras Hindu University)
वाराणसी/Varanasi-221005

DECLARATION BY THE CANDIDATE

I, RAKESH KUMAR, certify that the work embodied in this thesis is my own bonafide work and carried out by me under the supervision of Dr. AMRITA CHATURVEDI from JULY-2018 to DECEMBER-2023, at the DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, Indian Institute of Technology (BHU) Varanasi. The matter embodied in this thesis has not been submitted for the award of any other degree/diploma. I declare that I have faithfully acknowledged and given credits to the research workers wherever their works have been cited in my work in this thesis. I further declare that I have not willfully lifted up any other's work, paragraphs, text, data, results, etc., reported in journals, books, magazines, reports, dissertations, theses, etc., or available on websites and have not included them in this thesis and have not cited as my own work.

Date : 22/12/2023

Place : Varanasi

Rakesh

Signature of the Student
(RAKESH KUMAR)

CERTIFICATE BY THE SUPERVISOR

It is certified that the above statement made by the student is correct to the best of my/our knowledge.

Amrita

Signature of Supervisor

(Dr. AMRITA CHATURVEDI)

Department of Computer Science and Engineering
Indian Institute of Technology

(Banaras Hindu University)क/Supervisor

Varanasi-221005

संगणक विज्ञान एवं अभियंत्रण विभाग
Department of Computer Sc. & Engg.
भारतीय प्रौद्योगिकी संस्थान
Indian Institute of Technology
(काशी हिन्दू विश्वविद्यालय)
(Banaras Hindu University)
वाराणसी Varanasi-221005
भारत

Date : 22/12/2023

Place : Varanasi

Sanjay Singh
Signature of Head of Department

(Prof. Sanjay Kumar Singh)

आचार्य व विभागाध्यक्ष

Profesⁱⁱⁱ & Head

संगणक विज्ञान एवं अभियंत्रण विभाग

Department of Computer Sc. & Engg.

भारतीय प्रौद्योगिकी संस्थान

Indian Institute of Technology

(काशी हिन्दू विश्वविद्यालय)

(Banaras Hindu University)

COPYRIGHT TRANSFER CERTIFICATE

Title of the Thesis: SOME MACHINE LEARNING ASSISTED SOFTWARE BUG PREDICTION TECHNIQUES

Name of the Student: RAKESH KUMAR

Copyright Transfer

The undersigned hereby assigns to the Indian Institute of Technology (Banaras Hindu University) Varanasi, all rights under copyright that may exist in and for the above thesis submitted for the award of the DOCTOR OF PHILOSOPHY.

Date : 22/12/2023

Place : Varanasi


(RAKESH KUMAR)

Note: However, the author may reproduce or authorize others to reproduce material extracted verbatim from the thesis or derivative of the thesis for author's personal use provided that the source and the Institute's copyright notice are indicated.

To
My Beloved Family
Friends, Teachers
and
Lord Shiva

ACKNOWLEDGEMENTS

“Knowledge is in the end based on acknowledgment.” -Ludwig Wittgenstein

I want to take this opportunity to express my deep sense of gratitude to all who helped me directly or indirectly during this thesis work. Firstly, I would like to thank my supervisor, Dr. Amrita Chaturvedi, for being a great mentor and the best adviser I could ever have. Her advice, encouragement, and critics are a source of innovative ideas, inspiration, and causes behind the successful completion of this Thesis work. The confidence shown on me by her was the most significant source of inspiration for me. It has been a privilege working with her for several years. I am highly obliged to all the faculty members of the Computer Science and Engineering Department for their support and encouragement. I express my sincere thanks to RPEC members Dr. Lakshmanan Kailasam of the Department of Computer Science and Engineering and Dr. Ashok Ji Gupta, Department of Mathematical Sciences IIT (BHU), for providing continuous support, encouragement, and advice. I express my sincere thanks to all the Professors, office staff, supporting staff, and Ph.D. Research Scholars of Indian Institute of Technology (BHU) Varanasi India. I express my gratitude to Director, Registrars, Deans, Heads, and Student Alumni of the Indian Institute of Technology (BHU) Varanasi.

My memory of the study period at IIT (BHU) can never be complete without mentioning my fellow research scholars. Special thanks to Mr. Anurag Tiwari, Mr. Shashank Kumar Singh, Miss. Shruti Bajpai, Mr. Supriya Chanda, Ms. Naina Yadav, Manisha Singh, Miss. Divya Singh, Ms. Sheetal, Miss. Nitika Nigam, Mr. Sandeep Gautam, Mr. Shivansh Mishra, Mr. Ramakant Kumar, Dr. Santosh Tripathy, and Mr. Vinod Kumar for their great help and cooperation. I extend special thanks to the non-teaching staff in the department, particularly Mr. Ravi Kumar Bharti, Mr. Prakhar Kumar, and Mr. Shubham Pandey, for their consistent support.

My father, mother, and elder fathers/mothers, namely Mr. Umesh Yadav, Mr. Keshav Yadav, Mr. Vishnudhari Singh, Mr. Ram Niwas Singh, Ms. Sumitra Yadav, Ms. Surati Yadav, Ms. Savitri Yadav, and Lalmuni Yadav, who gave me the power and brain to work out on this research and their help at every level, made me see

this success. I owe thanks to my dearest wife, Ms. Archana Yadav for her continued and unfailing love, support and understanding during my pursuit of a Ph.D. degree that made the completion of my thesis possible. You all were always around at times when I thought that it is impossible to continue; you all helped me to keep things in perspective. I greatly value your contribution and deeply appreciate your belief in me.

Words are insufficient to express my profound sense of gratitude to my friends Rajkumar Yadav, Ashish Kumar, Sachin Mishra, Brijesh, Siddharth, Er. Satyendra Srivastava, Vikasanand, Dr. K.P. Yadav, Dr. R.H. Khan, Dr. Rajkumar Saini. My sisters and brothers Poonam Yadav, Sindhu, Bindu, Pooja, Anu, Tanu, Vimesh, Chandrakesh, Krishan, Ajit, Akhilesh, Arjun, Kamlesh, Jitendra, Mukesh, Dinesh and Rajesh, Raju, whose encouragement gave me physical and moral strength throughout my career as well in the present research. I extend my thanks to my father/mother-in-law, Mr. Chhotak Yadav, Ms. Sarita Yadav, brother-in-law Hari-mohan, and daughter Gargi, who are part of my inspiration. Finally, I would like to wind up by paying my heartfelt thanks and prayers to the Almighty for their unbound love and grace.

- Rakesh Kumar

Contents

List of Figures	xv
List of Tables	xvii
Abbreviations	xix
Symbols	xxi
Preface	xxiii
1 Introduction	1
1.1 Research Background	1
1.1.1 The Existence and Harm of Software Bugs	1
1.1.2 Software Bugs	3
1.1.3 Software Testing	3
1.2 Software Bug Prediction	4
1.2.1 Types of SBP Based on Labeled Software Modules	6
1.2.2 Category of SBP Based on Bug Values	7
1.3 Brief Introduction of studied SBP Scenarios	8
1.3.1 Classification-Based Supervised SBP	8
1.3.2 Classification-Based Unsupervised SBP	9
1.3.3 Regression-Based Unsupervised SBP	10
1.3.4 Classification-Based Unsupervised SBP in Functional Paradigm (FP)	11
1.4 Major Challenges Faced in SBP Scenario	12
1.5 Motivation for the Thesis	14
1.5.1 Motivation for Classification-Based Supervised SBP	14
1.5.2 Motivation for Classification-Based Unsupervised SBP	15
1.5.3 Motivation for Regression-Based Unsupervised SBP Prediction	17
1.5.4 Motivation for Classification-Based Unsupervised SBP in FP	19
1.6 Thesis Objective	21
1.7 Contributions to the Thesis	22
1.7.1 WMV: Classification-Based Supervised SBP	23
1.7.2 TCL/TCLP: Classification-Based Unsupervised SBP	24

1.7.3	MTB/MTBP: Regression-Based Unsupervised SBP	26
1.7.4	UMV: Classification-Based Unsupervised SBP in FP	27
1.8	Thesis Organization	29
2	Related Work and Preliminaries	31
2.1	Related Work on Supervised SBP	31
2.1.1	Classification-Based Supervised SBP	32
2.1.2	Regression-Based Supervised SBP	43
2.2	Related Work on Unsupervised SBP	46
2.2.1	Classification-Based Unsupervised SBP	46
2.2.1.1	Threshold-Based Unsupervised SBP	47
2.2.2	Regression-based Unsupervised SBP	48
2.3	Studies on Classification-Based SBP in FP	49
2.4	Preliminary	50
2.4.1	Software Metrics	50
2.4.2	Datasets Description	52
2.4.2.1	Benchmark Datasets	52
2.4.3	Performance Measures	56
2.4.3.1	Classification-Based Performance Measure	56
2.4.3.2	Regression-Based Performance Measure	58
2.4.3.3	Boxplot Presentation	59
2.4.4	Statistical Test	59
2.4.5	Baseline Methods	60
2.4.6	Hardware and Software Used	62
3	A Supervised SBP Technique Using Reward-Based WMV Ensemble Approach	63
3.1	Introduction	63
3.2	Proposed Approach: WMV	64
3.2.1	Simple Majority Voting Ensemble Method (SMV)	66
3.2.2	Weighted Majority Voting (WMV) Ensemble Method	67
3.2.2.1	Reward-Based Weight Evaluation Scheme	68
3.3	Experimental Design	72
3.3.1	Research Questions (RQ)	72
3.3.2	Datasets (DSs)	74
3.3.3	Experimental Baseline	74
3.3.4	Performance Measures	75
3.4	Experimental Results	75
3.4.1	Results with Respect to Accuracy	79
3.4.2	Results with Respect to FM	81
3.4.3	Results with Respect to MCC	83
3.4.4	Response of Research Questions	85
3.4.5	Result Discussion	92
3.5	Threats to Validity	93

3.6	Conclusion and Future Work	94
4	An Unsupervised SBP Technique Using Threshold Derivation	97
4.1	Introduction	97
4.2	Proposed Approach: TCL/TCLP	99
4.2.1	Threshold Derivation	100
4.2.1.1	Descriptive Statistics and Distribution	100
4.2.1.2	Software Metrics Threshold Calculation	101
4.2.2	Clustering and Labelling	101
4.2.3	Metric Selection and Instance Selection	103
4.2.4	Learning and Prediction	110
4.3	Experimental Design	112
4.3.1	Research Questions	112
4.3.2	Datasets (DSs)	113
4.3.3	Experimental Baselines	113
4.3.4	Performance Measures	114
4.4	Experimental Results	115
4.4.1	Results in Terms of Accuracy	120
4.4.2	Results in Terms of FM	122
4.4.3	Results in Terms of MCC	123
4.4.4	Results After Applying Sampling Technique	123
4.4.5	Answer of Research Questions	124
4.5	Explainability and Interpretability of TCL/TCLP	127
4.5.1	Explanation on Number of Metrics and Instances Selected for Training	127
4.5.2	Number of Data Points Correctly Labeled by Proposed Algorithm (TCLP) Differ from the Rest of the Methods within Each Group	128
4.5.3	Automated Labeling Versus Expert Labeling	130
4.5.4	Significance of Variation in Number of Metrics Selected Across Each DS Group Using MVR	131
4.5.5	Significance of Variation in Percentage of Instances Selected Across Each DS Group Using IVR	131
4.5.6	Execution Time of Different Models	133
4.6	Threats to Validity	133
4.7	Conclusion and Future Work	136
5	An Unsupervised SBCV Prediction Technique Based on Selected Software Metrics	137
5.1	Proposed Approach: MTB/MTBP	139
5.1.1	Unsupervised Metric Selection (UMS) Method	140
5.1.2	Threshold Calculation	143
5.1.3	Software Bug Count Vector (SBCV) Generation	144
5.1.4	Machine Learning Model Building	146

5.1.5	Software Bug Count Vector (SBCV) Prediction	147
5.2	Experimental Design	148
5.2.1	Research Questions	148
5.2.2	Datasets	149
5.2.3	Performance Measures	150
5.3	Experimental Results	150
5.3.1	Result Analysis for RQ1	151
5.3.2	Result Analysis for RQ2	156
5.4	Explainability and Interpretability of SBCV Prediction Models	159
5.4.1	Software Metrics Selected	159
5.4.2	Performance Analysis of MTB/MTBP and Baseline Models with respect to Bug %	159
5.4.3	Performance Analysis of MTB/MTBP and Baseline Models with respect to Software Projects Size	161
5.4.4	Usage Scenario and Limitations of MTB/MTBP	162
5.5	Threat to Validity	163
5.6	Conclusion and Future Work	164
6	An Unsupervised SBP Technique in Functional Paradigm	167
6.1	Introduction	167
6.2	Datasets Description	170
6.2.1	Dataset Collection	172
6.2.2	Software Metric Values Extraction:	174
6.3	Proposed Approach: UMV	175
6.4	Experimental Design	181
6.4.1	Research Questions	181
6.4.2	Performance Measure	183
6.4.2.1	Average Silhouette Width Coefficient	183
6.5	Experimental Results	184
6.5.1	Answer of Research Questions	191
6.5.2	Experimental Findings	191
6.5.3	Discussion	193
6.6	Threat to Validity	194
6.7	Conclusion and Future Work	195
7	Discussion	197
8	Conclusion and Future Work	203
8.1	Conclusions	203
8.2	Future Work	206
A	List of Publications	211
A.1	Journal Papers	211
A.2	Conference Papers	212

B Sample Datasets and Other Results	213
B.1 Distinguishing properties of functional paradigms	220
 Bibliography	 223

List of Figures

1.1	Bug class (SBP) VS. bug count vector prediction (SBCV)	19
1.2	Thesis structure	30
1.3	Concept map of thesis	30
3.1	Block diagram to develop the proposed SBP	65
3.2	Comparative performance of different techniques are shown by boxplots with respect to (a) Accuracy, (b) FM, and (c) MCC.	76
3.3	Critical diagram (using Nemenyi test) representation for the WMV and other techniques on 28 DSs with respect to (a) Accuracy, (b) FM, and (c) MCC.	77
3.4	Pairwise boxplot comparison of different recent approaches with the proposed approach WMV	91
4.1	Block diagram of the proposed approach showing the steps involved in TCL/TCLP	99
4.2	Boxplot performance of different approaches in term of (a) Accuracy, (b) FM, and (c) MCC	120
4.3	Performance analysis of proposed approach and other approaches using Post Hoc Nemenyi Test conducted on 28 DSs in terms of (a) Accuracy, (b) FM, and (c) MCC	121
4.4	Performance of TCLP approach with variation in the number of metrics selected in terms of (a) Accuracy, (b) FM, and (c) MCC	132
4.5	Performance of TCLP approach with variation in percentage of instances selected in terms of (a) Accuracy, (b) FM, and (c) MCC	134
5.1	Step-by-step block diagram illustrating the development process of the proposed SBCV prediction model (MTB/MTBP)	139
5.2	Coverage measure versus added metrics for Ant-1.6 dataset	143
5.3	Boxplot performance of MTB/MTBP and baseline models in terms of (a) MAE, (b) MRE, and (c) Pred(l)_Error	152
5.4	Performance analysis of MTB/MTBP and baseline models using Post Hoc Nemenyi Test in terms of mean score (a) MAE, (b) MRE, and (c) Pred(l)_Error	157
5.5	Average performance analysis of MTB/MTBP and other baselines with the number of selected metrics conducted on 22 datasets in terms of (a) MAE, (b) MRE, and (c) Pred(l)_Error	158

5.6	Performance analysis of MTB/MTBP and baseline models with respect to bug % in 22 datasets in terms of (a) MAE, (b) MRE, and (c) Pred(1).Error	161
5.7	Performance analysis of MTB/MTBP and baseline models with respect to software projects size in terms of (a) MAE, (b) MRE, and (c) Pred(1).Error	163
6.1	Block diagram of the proposed approach showing the steps involved in UMV	170
6.2	Block diagram for step-by-step procedure to extract software metrics from the Haskell Packages	174
6.3	Plots showing the Average silhouette width vs. number of clusters k in the four datasets (a) Ethereum (b) Hburg (c) Mios (d) Piet	179
6.4	Boxplot presentation of the different models in terms of (a) accuracy, (b) f-measure, and (c) MCC	187
6.5	Critical diagram presentation of the different models in terms of (a) accuracy, (b) f-measure, and (c) MCC	188
6.6	Average silhouette plot for clusters generated by UMV techniques for four Haskell packages (a). Ethereum, (b) Hburg, (c) Mios, (d) Piet	189
6.7	Defect-wise boxplot to represent distribution of software metric values of defective and non-defective instances in each DS for clusters generated by UMV technique on four Haskell packages (a). Ethereum, (b) Hburg, (c) Mios, (d) Piet. Red boxplot (first) is for non-defective functions and blue boxplot (second) is for defective functions	192

List of Tables

2.1	Summary of SBP methods and metrics used by different researchers	33
2.2	Group-wise list of original software metrics in different datasets	53
2.3	Classification based 28 Software projects collected from five groups	54
2.4	Regression-based 22 benchmark datasets collected from two groups	55
2.5	Confusion matrix to classify whether a class is buggy or not	57
3.1	Prediction probability of base classifiers for each instance	67
3.2	Weights of each classifier evaluated after traversing each instance of testing dataset	69
3.3	An example dataset with predicted probability ($p[i, k]$), corresponding class label using BCs and original class label	70
3.4	Change in weights of base classifiers across each instance	72
3.5	Weighted probability $\mathcal{P}[i, k]$ calculation and original class label	72
3.6	Performance of different SBP techniques with respect to accuracy	80
3.7	Wilcoxon signed-rank test analysis ($p < 0.05$): Mean difference (Column-Row) between the Accuracy of different approaches and p-value. Mean difference: Left bottom triangle; p-value: Right upper triangle	81
3.8	Performance of different SBP techniques with respect to F-measure	82
3.9	Wilcoxon signed-rank test analysis ($p < 0.05$): Mean difference (Column-Row) between the F-measure of different approaches and p-value. Mean difference: Left bottom triangle; p-value: Right upper triangle	83
3.10	Performance of different SBP techniques with respect to MCC	84
3.11	Wilcoxon signed-rank test analysis ($p < 0.05$): Mean difference (Column-Row) between the MCC of different approaches and p-value. Mean difference: Left bottom triangle; p-value: Right upper triangle	85
3.12	Results of state-of-the-art (SOTA) techniques and WMV on each dataset in terms of FM	87
3.13	Average performance in terms of different metrics, Win-Tie-Loss value of WMV as compared to SOTA techniques in terms of FM over all datasets, Mean_Difference (WMV-SOTA), and p-value	90
4.1	An example for an unlabeled DS to labeled based on metric threshold	102
4.2	Finding the value of Z for each instance of unsupervised DS	105
4.3	Example for an unlabelled dataset to labeled dataset	105
4.4	Dataset after labeling and metric violation ratio	106
4.5	Dataset after selected metrics and instance violation ratio	107

4.6	Dataset after selected metrics and instances (Training DS)	108
4.7	Performance of different SFP techniques in terms of accuracy	116
4.8	Performance of different SFP techniques in terms of FM	117
4.9	Performance of different SFP techniques in terms of MCC	118
4.10	Wilcoxon signed-rank test analysis ($p < 0.05$): Mean difference and p-value among the performance of different approaches	119
4.11	Performance of TCLP without sampling and with sampling (SMOTE) technique	126
4.12	Selected metrics and number of instances from each project after filtering	129
4.13	Difference in correctly labeled data points between the proposed algorithm (TCLP) and rest of the 13 methods within each group (Other method-TCLP)	130
4.14	Execution time of different SBP models in seconds	135
5.1	BCV generation using software metric threshold example	145
5.2	Results of MTB/MTBP and 8 machine learning models in terms of MAE, MRE, and Pred(l) Error	154
5.3	Effect size using Cohen's D (shown in Right Upper Triangle) and p-value using Wilcoxon signed-rank test (shown in left-bottom triangle) among all the models	155
5.4	Top selected metrics from each dataset after UMS and name of all metrics used group-wise	160
6.1	Fundamental difference between the proposed approach UMV and other existing threshold-based SDP methods	169
6.2	Statistical description of the SMs extracted from Haskell Packages	173
6.3	Performance of different models in terms of accuracy, f-measure, and MCC	186
B.1	Sample bug dataset (First 20 modules of Tomcat project)	213
B.2	Eclipse and Android bug datasets descriptions	214
B.3	Performance of SBP techniques (Chapter 3) over Eclipse and Android datasets in terms of accuracy, FM, MCC	215
B.4	Performance of different SBP techniques (Chapter 4) over Eclipse datasets (larger datasets)	216
B.5	Results of MTB/MTBP and 8 ML models (Chapter 5) over Eclipse and Android DSs	217
B.6	Effect size using CohenD between WMV and others models (Chapter 3)	218
B.7	Effect size using CohenD between TCLP and others models (Chapter 4)	218
B.8	Effect size using CohenD between UMV and others models (Chapter 6)	218
B.9	Cross-version (Training on previous version and testing on current version) performance of different SBP models (Chapter 3)	219

Abbreviations

ACL	A verage C lustering L abel
BCs	B ase C lassifiers
BRR	B ayesian R idge R egression
CLAMI	C lustering L abelling M etric selection I nstance selection
DTR	D ecision T ree R egression
DSs	D atasets
FCV	F old C ross V alidation
FM	F - M eaure
FP	F unctional P aradigm
GA	G enetic A lgorithms
HC	H ierarchical C lustering
IVR	I nstance V iolation R atio
IVS	I nstance V iolation S core
ITV	I nstances T hreshold V iolation
KMS	K -means
LM	L inear R egression
MAE	M ean A bsolute E rror
MBC	M odel B ased C lustering
MCC	M atthew's C orrelation C oefficient
ML	M achine L earning
MLP	M ulti L ayer P erceptron
MTB	M etric selection T hreshold derivation B ug count vector

MTBP	MTB Plus
MTV	Metric Threshold Violation
MRE	Mean Relative Error
MVS	Metric Violation Score
NBR	Negative Binomial Regression
NGC	Neural Gas Clustering
NB	Naive Bayes
OOP	Object-Oriented Paradigm
Pred(<i>l</i>)_Error	Prediction at level <i>l</i> Error
RD	Reduced Dataset
RF	Random Forest
ROSE	Random Oversampling Examples
RQ	Research Question
RUS	Random Undersampling
SD	Standard Deviation
SM	Software Metrics
SMOTE	Synthetic Minority Oversampling Technique
SMV	Simple Majority Voting
SBCV	Software Bug Count Vector
S[D, B, F]P	Software [Defect, Bug, Fault] Prediction
TCL	Threshold Clustering Labelling
TCLP	TCL Plus
UMS	Unsupervised Metric Selection
UMV	Unsupervised Majority Voting

Symbols

ω_k	Weight of k^{th} classifier
η	Number of BCs
n	Number of instances
$p_{i,k}$	Prediction probability for i^{th} instance by k^{th} classifier
ϕ	BC function
m	Number of metrics
$\hat{y}_{i,k}$	Prediction label vector
$\delta_{i,SMV}$	Resultant label vector using SMV method
i	For each instance i
β_i	Ratio of the number of wrong BC (Z) and η
Z_i	Number of wrong predictors for i^{th} instance
$\omega_{i,k}$	Weight of k^{th} BC after predicting label of the i^{th} instance
$\mathcal{P}_{i,k}$	Weighted probability
θ_k	Threshold
$\lambda_{i,k}$	Weighted label
$\delta_{i,WMV}$	Predicted class label using WMV
$\ln(x)$	Log function
μ_m	Mean
Ω_m	Standard deviation
T'_m	Symbolic metric threshold
T_m	Original threshold

$exp()$	Exponential function
X_k	Metric vector
δ_i	Minimum Euclidean distance
Ψ_i	Coverage measure
$\bar{\delta}$	Mean of δ_i
Φ	Effect size magnitude
γ_m	Skewness index
C	Number of clusters
U	n number of data points
$S(i)$	Silhouette coefficient

PREFACE

Software is now a part of every aspect of our daily lives. But, when we design, develop, and configure software, bugs are inevitable. These hidden bugs in the code snippet make the software insecure and unreliable. So, it's crucial to find and fix these bugs before we deliver the software to the end user. However, because software is getting larger and more complex, it's becoming a challenging task for software developers and testers to improve the software quality.

Software bugs occur when software doesn't meet the requirements provided in the software requirement specification (SRS) or end users' expectations (even if they aren't explicitly stated in SRS but are reasonable). A software bug prediction (SBP) model predicts bugs in each module of a software system. Once we predict where the bugs might be, the software quality assurance (SQA) team can optimally assign limited testing resources. This means more testers are required for the modules that have a high probability of containing bugs. This approach helps to save time and money by reducing testing efforts and software development costs.

Many software researchers have developed several SBP techniques to predict modules as buggy or non-buggy, utilizing statistical techniques, machine/deep learning, and ensemble learning approaches. These SBP techniques have many challenges, viz. missing values, data redundancy, irrelevant features, correlations, and class imbalance problems. The skewed distributions of software metrics in datasets may lead to issues like overfitting and dataset quality concerns.

The central aim of this thesis is to illuminate the advantages of software metrics (SMs) selection, machine learning (ML), or ensemble learning (EL) in predicting buggy/ non-buggy modules or bug counts in each software module using supervised and unsupervised SBP approaches. It also strives to mitigate class imbalance and

overfitting problems to produce unbiased results. Furthermore, building the supervised SBP model needs labeled datasets, so an alternate, an unsupervised SBP model is also proposed. Additionally, predicting only buggy and non-buggy modules may not help to categorize the modules and can not provide bug count information. So, to overcome these, an unsupervised bug count vector prediction model is proposed. The aforementioned supervised/unsupervised SBP models are employed on the object-oriented paradigm (OOP) or imperative programming datasets. These datasets are created from JAVA, C, and C++ software projects and are publicly available to software researchers. Additionally, we could not find any software bug dataset for Haskell and functional paradigm (FP). Therefore, to explore FP, four novel FP bug datasets are created using Haskell, which is an FP language. Further, a new classification-based unsupervised SBP model is proposed to be employed on novel FP datasets.

Based on the aforementioned discussion, the proposed work in this thesis is categorized into four contributing chapters. Chapter 3 adopts a classification-based supervised SBP approach. Then, in Chapter 4, we present a classification-based unsupervised SBP approach. Furthermore, in Chapter 5, we propose a regression-based unsupervised SBP approach to predict bug count vectors in each module. Additionally, Chapter 6 describes the creation of novel FP datasets and a novel classification-based unsupervised SBP model employed on FP datasets. The primary focus of this thesis is to explore the applications of ML techniques to address the specific challenges encountered in these four different SBP scenarios. The ultimate goal of this thesis is to enhance the overall performance of the existing state-of-the-art SBP models. The proposed four SBP approaches in this thesis are summarized as follows:

In order to enhance the performance of supervised ML/ ensemble learning SBP techniques and handle data imbalance problems, we present a novel classification-based SBP approach called reward-based weighted majority voting (WMV). In WMV, each base classifier's (BC's) performance is assessed, and a reward-based

mechanism is used to assign different weights to each classifier. BCs that make correct predictions receive rewards, and there are no penalties for wrong predictions. BCs that correctly predict more instances get higher weights in the ensemble. WMV surpasses the performance of the mentioned BCs, simple majority voting (SMV), and many recent state-of-the-art techniques in terms of accuracy, F-Measure, and Matthew’s correlation coefficient (MCC).

The previously proposed SBP technique WMV employs on labeled datasets. To overcome the need for labeled datasets, handle the imbalanced datasets, and enhance the performance of unsupervised ML SBP techniques, we’ve developed a novel classification-based SBP approach called TCL/TCLP (Threshold Clustering Labelling/Threshold Clustering Labelling Plus). TCL/TCLP doesn’t rely on the labeled dataset. This approach is based on the existing CLAMI technique but enhances it using logarithmic transformation and deriving metric thresholds. TCL labels instances into binary classes (buggy or non-buggy) based on the corresponding threshold of software metrics (SMs). TCLP extends TCL one step further by performing metrics/instances selection and ML training using a random forest algorithm to build an SBP model. Our empirical evaluation on 28 datasets from five software groups, varying in metrics and granularity, demonstrates that this unsupervised SBP method outperforms state-of-the-art techniques. It significantly improves accuracy, F-measure, and MCC compared to the majority of existing approaches.

The above proposed SBP technique WMV and TCL/TCLP can predict only binary classes, viz. buggy or non-buggy. So, to extend this study, we have proposed a Software Bug Count Vector (SBCV) prediction technique. It is a regression model that focuses on predicting the precise number of bugs in each module of a software system, which can help to optimize resource allocation and software maintenance. Most of the previous researches have relied on labeled datasets to predict SBCV using regression algorithms, but collecting and labeling the datasets are challenging tasks. To address this issue and the lack of an unsupervised regression model for bug count prediction, we introduce a novel unsupervised regression model (SBCV). Our

approach predicts SBCV on unlabeled data by determining independent thresholds for each software metric. Our method outperforms many standard supervised algorithms on 22 datasets in terms of mean absolute error (MAE), mean relative error (MRE), and $\text{Pred}(l)\text{-Error}$. Considered statistical tests confirm the significance of our approach.

Finally, the above-proposed techniques are employed only on OOP datasets. So, to overcome this issue, we explore SBP for Haskell, a functional programming language that hasn't been extensively investigated before. Our first objective is to create novel datasets by extracting source code metrics from Haskell functions. As the second objective, we utilize statistical analysis to determine the right-skewed nature of these metrics and hence apply three transformation techniques to reduce the skewness and calculate thresholds. Then, we propose an unsupervised majority voting (UMV) ensemble method to develop a classification-based SBP model and compare its performance. Our experimental results show that UMV performs well in Haskell, with high accuracy, f-measure, and MCC. This approach can be valuable for predicting software reliability in the absence of a labeled dataset.

In conclusion, this thesis addresses challenges in various SBP scenarios and proposes new approaches to enhance prediction performance using ML techniques. It extends the use of ML in software engineering and offers innovative solutions for SBP that benefit SQA activities.