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 Pred(l)_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.