Chapter 7

Summary

This thesis investigated and suggested four different defect prediction approaches in four distinct aspects. This thesis also discussed the various challenges of defect prediction and how our methods overcome these challenges. We have revisited existing solutions to these challenges and also analyzed them. WWe proposed a classification-based SDP model. We have also extended the classification-based SDP problem into regression-based defect estimation methods in three different categories. The first is cross-version defect number prediction, the second is cross-project defect count estimation, and the third is for entire software prediction, which is hybrid regression analysis. The summary of all four works is given below.

1. BPDET is a classification-based defect prediction method. It employed a stacked denoising autoencoder to extract deep features from the classical software metrics of 12 NASA projects in the deep learning phase. After that, it uses heterogeneous ensemble learning techniques to train the model; ten different classifiers are used as a weak learner and average probability as a combination rule in the ensemble learning phase. According to our empirical study, the deep learning phase mainly addresses the class imbalance and over-fitting problem, and the two stages of ensemble learning also target these issues. We found that BPDET significantly outperformed the existing state-of-the-art methods in terms of MCC, ROC, PRC, and F-measure. We conducted a k-fold cross-validation test to measure the stability of the model, and we found BPDET is highly stable.

2. Cross-version defect count vector prediction estimates the bug count vector (column vector consists of bug information) of the next version of a software system. It mainly employs the information of all prior versions of the same software system. BCV-Predictor is an approach that estimates the bug count vector of the next version software system. It consists of two main steps; first, the data amalgamation process that leads to meta data creation, and second, model training. We have employed long short term memory architecture as a learning technique. We trained the model using meta data and estimated the bug count vector over the test/development set. We found the proposed model produces unbiased results over existing techniques and avoid overfitting and class imbalance problem. Moreover, the proposed model significantly outperforms existing methods. The BCV-Predictor is more suitable for large software projects, and it takes reasonable training costs. The model is stable for substantial projects and moderately stable for a small software system.

3. Cross-project defect number prediction is one step ahead of cross-project defect prediction. It also estimates the number of faults in every module in the target project. We proposed a method called DNNAttention using attention and LSTM layer to predict a number of the defect in the target project using training over diverse projects (cross-project). The DNNAttention contains two phases. In the first phase, the data is created (source project) using the data amalgamation process; it uniquely amalgamates 44 different projects. In the second phase, deep features are extracted using the attention layer and train the model using the LSTM network. We train the proposed model using 44 diverse projects of the PROMISE repository and test over the individual project (target project). The results found the proposed model significantly outperforms existing methods. It also avoids class imbalance and overfitting problems. We discovered that DNNAtention is more suitable for large and moderate-sized projects, whereas it produces a high loss for small-sized projects. The model is stable over large and moderate size projects.

4. Prediction of the entire version of a software system will reduce the enormous amount of effort in software development. We have proposed a novel approach to estimate the entire next version of a software system. In the hybrid regression analysis, we applied deep learning and eight projects of the PROMISE repository for the experiments. Our approach is divided into two phases: the first is the data augmentation phase, and the second is the entire

version prediction phase. The proposed model significantly outperforms other state-of-the-art methods.

The proposed methodologies can help to produce high-quality software products at a lower cost.