CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

In the growing scenario, software miners have developed numerous modules for defect prediction replacing the earlier existing conventional statistical approaches. Basically, the classifier model or the predictor model is designed in a manner to output two cases: one is the defect prone software and the other is the defect free software. To start the process of classifier models, the already known instances will be given as input to the classifier models. On the effective training of the proposed model, this will be tested with that of the available unknown instances and then the prediction will be carried out. Several studies have been carried out in the early literature based on the performance metrics (Han & Jing 2015) and design requirements (Zhang et al. 2014). This chapter presents the various related works available in the early literatures in the considered area of research on software defect prediction during the software development process.

2.2 BACKGROUND ON SOFTWARE PREDICTION MODELS

There exist several statistical and machine learning methods to identify defects in the newly developed software modules. A hybrid instance selection using nearest – neighbour (Ryu Duksan et al. 2015), cost- sensitive
boosting neural networks (Ryu Duksan et al. 2015), Fuzzy linear regression model (Valles-Barajas 2015), hybrid artificial neural network with artificial bee colony (Arar Ömer Faruk & Kürşat Ayan 2015), kernel ensemble learning approach (Wang et al. 2015) were few of the algorithms employed for this problem. The major issues of software defect prediction was identified and studied by Arora et al. (2015).

Apart from these above said methods, several other prediction models were developed and applied for the open source NASA datasets available at the PROMISE repository (He et al. 2015, Gabriela Czibula et al. 2014, Jindal et al. 2014, Arvinder & Kamaldeep 2015, Wang et al. 2015, Abaei Golnoush et al. 2015, Czibula Gabriela et al. 2014, Wahono et al. 2014, Catal Cagatay 2014, Liu Mingxia et al. 2014, Wahono et al. 2014, Kaur Amardeep & Kanwalpreet Kaur 2014, Askari et al. 2014). Han and Jing (2015) employed a high computational wrapper model with a significant improvement in recall rate and F-measure. In a similar way, ensemble decision trees and CART were also employed for performing cost-sensitive classification for SDP (Siers Michael & Zahidul Islam 2015,). A Bayesian Regularization (BR) approach is employed to determine the software faults along with Levenberg – Marquardt algorithm and Back Propagation algorithm (Olivier Vandecruys et al. 2008). A Tabu Search Fault Localization with Path Branch and Bound procedure on Software Engineering (TSFL-PBB) was employed to overcome the defect on fault localization (Pandiyan & Krishnakumari 2015). Multistage model for software defect density indicator employing the topmost reliability-relevant metrics and fuzzy inference system (FIS) was proposed by Yadav et al. (2015). A multi objective defect predictor (MODEP), based on multi-objective forms of machine learning techniques - logistic regression and decision trees specifically - trained using a genetic
algorithm for cross-project prediction, based on local prediction upon clusters of similar classes (Canfora Gerardo et al. 2015).

Data mining approach was employed to show the attributes that predict the defective state of software modules and is used in large software projects to detect defective modules that will cause failures during the software execution process (Yousef Ahmed 2015).

An iterative feature selection approach which repeatedly applies data sampling (to overcome class imbalance) followed by feature selection (to overcome high dimensionality), and finally combines the ranked feature lists from the separate iterations of sampling has been applied for several groups of datasets from two real-world software systems and used two learners to build classification models (Gao kehan et al. 2014, Han et al. 2014). The predictive ability of the evolutionary computation and hybridized evolutionary computation techniques for defect prediction was applied for datasets from the Apache Software Foundation using the Defect Collection and Reporting System (Malhotra Ravish et al. 2014). Zhang Wei et al. (2014) analyzed 44 metrics of application level, file level, class level and function level, and do correlation analysis with the number of software defects and defect density, the results show that software metrics have little correlation with the number of software defect, but are correlative with defect density.

Software defect prediction model was presented in the early literature for consecutive software products based on entropy and the process starts when the defect is found and ends when the resolution is verified and the defect is closed. Support Vector Machine (SVM) has been developed for software defect prediction using different kernels. Software defect prediction helps improve software quality by building effective predictive classification models using software metrics to facilitate identification of fault-prone modules (Thangavel & Nasira 2014).
Based on Genetic Algorithm has been developed for Software Defect Prediction and is been applied for datasets from the repositories (Wahono et al. 2014). A multistage model for software defect density indicator using the top most reliability relevant metrics and Fuzzy Inference System (FIS) has been developed for effective decision-support (Yadav et al. 2014).

The ability of requirement metrics for software defect prediction has been carried out employing six machine learning algorithms on the requirement metrics, design metrics, and combination of both metrics (Ma et al. 2014). A complete description on the summary of software prediction models over various period of study has been proposed by Han et al. (2015). Random forest algorithm based software prediction model developing an ensemble classifier was applied for large – scale software system (Suma et al. 2014).

From the above discussed literature reviews, it is inferred that the early proposed prediction models has not taken into account the misclassification cost of the non-defective and defective modules in large except in few cases (Ryu Duksan et al. 2015, Arar Ömer Faruk & Kürşat Ayan 2015, Liu Mingxia et al. 2014). Considering the real world problems, the rate of misclassification of defective module is highly important than that of the rate of misclassification of non-defective modules. The levels of these misclassifications are defined by their associated cost factors. Thus there are few efforts made in exploring the associated costs employing neural network architectures employing sampling procedures and threshold levels (Ryu Duksan et al. 2015). The variation is made in the threshold level of the neural network which decides the output until an optimal point is reached with respect to the cost matrix. From Zhou & Liu (2006), it is well noted that the movement of threshold is an appropriate factor to build cost-sensitive neural network architecture.
Rana et al. (2016) explored the defect inflow distribution of large software projects/release from two industrial domain and open source community. Also evaluated how information about defect inflow distribution from historical projects is applied for modeling the prior beliefs/experience in Bayesian analysis which is useful for making software defect predictions early during the software project lifecycle.

Verma & Kumar (2016) proposed predication of software defects using clustering approach with the help of static code metrics.

Liu et al. (2016) proposed a fuzzy integral method based on the mutual information (MIFI) to predict the software defect. Zhang et al. (2016) proposed a non-negative sparse graph based label propagation approach for software defect classification and prediction, which uses not only few labeled data but also abundant unlabeled ones to improve the generalization capability.

Shi et al. (2016) proposed a software reliability prediction approach based on software metrics. The proposed prediction approach could be applied using a variety of software metrics at different stages of the software development life cycle and could be used as an indicator of software quality.

Nam & Kim (2016) proposed a novel approaches, CLA and CLAMI that show the potential for defect prediction on unlabeled datasets in an automated manner without need for manual effort. Li et al. (2016) employed a modified bat algorithm, named Changing Range Bat Algorithm (CRBA) to optimize the parameters of Support Vector Machine (SVM) model. The classification ability of hybrid CRBA-SVM model surpasses all other approaches for software defect prediction. Cheng et al. (2016) proposed a novel weighted naive Bayes transfer learning algorithm. The proposed method
significantly improves cross-project defect prediction performance, compared to other methods.

Wang et al. (2016) proposed a defect prediction modeling framework based on multi-metric space and multi-type learning models. The model effectively improves the generalization performance and the predictive accuracy in software defect prediction. Hein & Saiedian (2016) predicted attack prone software components using repository mined change metrics.

The experiments over the Eclipse bug dataset demonstrate promising results in anticipation of the software defects. Cao et al. (2015) proposed an effective method called Transfer Component Analysis Neural Network (TCANN), by adequately considering the noise data, the class imbalance in data settings and transfer learning among cross-project. The proposed method is applied in software defect prediction.

Jindal et al. (2015) developed a model based on text mining techniques using machine learning method namely, Radial Basis Function of neural network. The proposed method is used for effective resource allocation and decision-making in software maintenance effort prediction. Valles-Barajas (2015) presented a comparative analysis between two techniques (fuzzy linear regression and statistical linear regression) to perform software defect estimation.

Gao et al. (2015) investigated two filter-based feature subsets selection techniques, i.e., correlation-based and consistency-based subset evaluation methods, and three data sampling methods, i.e., random under sampling, random oversampling, and synthetic minority oversampling. Abdi et al. (2015) presented a rule-based prediction approach based on kernel k-means clustering algorithm and Distance based Multi-Objective Particle Swarm
Optimization (DSMOPSO). The proposed method is used in software defect prediction.

Gao et al. (2015) investigated three data preprocessing approaches, in which feature selection is combined with data sampling, to overcome these problems in the context of software quality estimation. Wang et al. (2015) proposed feature subsets selection using fifteen feature selection methods and use newly proposed Average Pairwise Tanimoto Index (APTI) to evaluate the stability of the feature selection methods.

Caglayan et al. (2015) investigated the relationships between defects and test phases in order to build defect prediction models for different test phases. The proposed model gives a more granular outcome in terms of predicting defect-prone modules in each testing phase so that managers may better organize the testing teams and effort. Arar & Ayan (2015) proposed a different classification approach by combination of traditional Artificial Neural Network (ANN) and the novel Artificial Bee Colony (ABC) algorithm. The experimental results showed that a cost-sensitive neural network can be created successfully by using the ABC optimization algorithm for the purpose of software defect prediction.

Wang et al. (2015) proposed a multiple kernel ensemble learning (MKEL) approach for software defect classification and prediction. The experimental results show that MKEL outperforms several representative state-of-the-art defect prediction methods. The proposed method demonstrates the effectiveness of directly optimizing the model performance measure for the learning-to-rank approach to construct defect prediction models for the ranking task.

Shanthini & Chandrasekaran (2015) addressed ensemble approach of Support Vector Machine (SVM) for fault prediction. The
proposed method is superior to individual approach for software fault prediction in terms of classification rate through Root Mean Square Error Rate (RMSE), AUC-ROC, ROC curves.

He et al. (2015) validated the feasibility of the predictor built with a simplified metric set for software defect prediction in different scenarios, and to investigate practical guidelines for the choice of training data, classifier and metric subset of a given project.

Kaur & Kaur (2015) contributed to the application of machine learning in software defect prediction by investigating the robustness and stability of 17 classifiers on 44 open source software defect prediction data sets obtained from PROMISE repository.

Siers & Islam (2015) proposed a cost-sensitive classification technique called CSForest which is an ensemble of decision trees. Also proposed a cost-sensitive voting technique called CSVoting in order to take advantage of the set of decision trees in minimizing the classification cost. The proposed method is used in the research on software defect prediction. Jindal et al. (2015) developed a model based on text mining techniques that will be used to assign the severity level to each defect report based on the classification of existing reports done using the machine learning method namely, Radial Basis Function of neural network.

Han & Jing (2015) studied the relevance among different software metrics, and the software metrics and the decision of the software modules is measured by Mutual Information (MI), followed by a minimum information loss criterion. The proposed method has significant improvement in recall rate and F-measure, and is effective and efficient for software defect prediction.
Rana et al. (2015) identified possible directions for software defect prediction techniques and application in these domains by review the software development and testing process for two large companies from the automotive and telecom domain and map different defect prediction methods and their applicability to their lifecycle phases. Arora et al. (2015) distinguished some of the major issues of SDP and solution addressed them so far.

Rana et al. (2015) presents an association mining based approach that allows the defect prediction models to learn D modules in imbalanced datasets. The results show that the algorithm can be used in software defect prediction. Jayaraj et al. (2015) proposed a cubic spline based improved activation function for Multi-Layer Perceptron Neural Networks to classify defects based on software metrics. The proposed method can be used in software defect prediction.


Laradji et al. (2015) demonstrated the positive effects of combining feature selection and ensemble learning on the performance of defect classification. The proposed combined learning model resulted in remarkable classification performance paving the way for successful quality control in Software defect prediction.

Rubinic et al. (2015) reported preliminary results obtained by using a MATLAB variant of NSGA-II in combination with four simple voting strategies on three subsequent releases of the Eclipse Plug-in Development Environment (PDE) project. Preliminary results indicate that the voting procedure might influence software defect prediction performances. Mahajan et al. (2015) Bayesian Regularisation (BR) technique has been used for finding
the software faults before the testing process. The results signify that the software fault prediction model using BR technique provide better accuracy than Levenberg-Marquardt (LM) algorithm and Back Propagation (BPA) algorithm.

Jindal et al. (2015) applied text mining techniques to identify the relevant attributes from defect reports and relate these relevant attributes to software maintenance effort prediction. The proposed model is validated using 'Camera' application package of Android Operating System. The proposed method can be used for model prediction.

Rajaganapathy & Subramani (2015) surveyed about various techniques of fault prediction, clustering and classification to identify the defects in software modules. The experimental results confirm that the developed approach is applicable to software threat risk estimation.

Canfora et al. (2015) propose an approach coined as Multi-Objective Defect Predictor (MODEP) based on multi-objective forms of machine learning techniques - logistic regression and decision trees specifically - trained using a genetic algorithm. MODEP outperforms an alternative approach for cross-project prediction, based on local prediction upon clusters of similar classes in defect-prediction problem. Catal (2014) evaluated four semi-supervised classification methods for semi-supervised defect prediction. Zhang et al. (2014) analyzed 44 metrics of application level, file level, class level and function level, and do correlation analysis with the number of software defects and defect density, the results show that software metrics have little correlation with the number of software defect, but are correlative with defect density.

Liu et al. (2014) proposed a new Two-Stage Cost-Sensitive learning (TSCS) method for SDP, by utilizing cost information not only in the
classification stage but also in the feature selection stage. Experimental results suggest that our TSCS method achieves better performance in software defect prediction compared to existing single-stage cost-sensitive classifiers. Rana et al. (2014) advocated the use of Information Gain (IG) based technique to reduce size of input space by dropping the irrelevant metrics. The results show that use of IG improves the models’ performances more often than PCA does in software defect prediction. Siers & Islam (2015) proposed a cost-sensitive classification technique called CSForest which is an ensemble of decision trees. Also proposed a cost-sensitive voting technique called CSVoting. The initial experimental results indicate a clear superiority of the proposed techniques over the existing ones in software defect prediction.

Pushphavathi et al. (2014) introduced a novel hybrid method of Random Forest (RF) and Fuzzy C Means (FCM) clustering for building defect prediction model. The experimental results show that the proposed method provides a higher accuracy and a relatively simple model enabling a better prediction of software defects. Askari & Bardsiri (2014) proposed a new method by developing new learning methods based on support vector machine principles and using evolutionary algorithms. Results reveal that the proposed algorithm provides higher accuracy and precision compared to the other models in software defect prediction.

Lu et al. (2014) investigated the performance of semi-supervised learning for software fault prediction. The results show that the dimension-reduction with semi-supervised learning algorithm performs significantly better than one of the best performing supervised learning algorithm - random forest - in situations when few modules with known fault content are available for software defect prediction. Li et al. (2014) proposed the fuzzy measure and fuzzy integral to the classification of defects. Experimental results have shown that there are interactions between characteristic attributes of software
modules, and also proved that the fuzzy integral fusing method using Fuzzy Measure based on Genetic Algorithm (GA-FM) can significantly improve the accuracy for software defect prediction.

Ma et al. (2014) presented an improved semi-supervised learning approach for defect prediction involving class imbalanced and limited labeled data problem. Experimental results also show that with the proposed learning approach, it is possible to design better method to tackle the class imbalanced problem in semi-supervised learning in software defect prediction. Yadav & Yadav (2014) proposed a multistage model for software defect density indicator using the top most reliability relevant metrics and Fuzzy Inference System (FIS). This proposed model can be used in software defect prediction.

Khoshgoftaar et al. (2014) propose an iterative feature selection approach, which repeated applies data sampling followed by feature selection, and finally we perform an aggregation step which combines the ranked feature lists from the separate iterations of sampling. The proposed iterative feature selection approach outperforms the non-iterative approach in software defect prediction. Jindal et al. (2014) intended to predict a model based on text mining and machine learning technique which is used to assign a severity level to each of the defect found during testing. The proposed KNN method works best for medium severity defects as compared to the other severity defects in software defect prediction.

Considering the literature studies, numerous works have been carried out in the area of software defect prediction employing different techniques which includes – decision trees (Siers Michael et al. 2015, Yong et al. 2014) boosting models (Ryu et al. 2015), fuzzy predictor models (Yadav et al. 2014, Valles-Barajas & Fernando 2015, Li et al. 2014), support vector machine (Thangavel & Nasira 2014), neural predictors (Jindal et al. 2014, Abaei Golnous et al. 2015, Kaur et al. 2014, Askari et al. 2014, Shepperd et al.
2014, Wahono et al. 2014, Zhou & Liu 2006), bayes models (Okutan et al. 2014), rule based approaches (Abdi et al. 2015, Gabriela Czibula et al. 2014) and so on. A review on the various open issues in software prediction has been presented in Arora et al. (2015).

2.3 BACKGROUND ON ASSOCIATION RULE MINING FOR DEFECT PREDICTION MODELS

This section briefly describes the various studies done in the field of the association rule mining, relationships between object oriented metrics and its impact in defect prediction models. Association rule mining (Baralis et al. 2012) identifies interesting relationships among data items and is used as unsupervised learning scenarios. The discovery of these relationships helps in many decision making processes. Association rule mining finds those rules that satisfy some minimum support and minimum confidence constraints. Several extensions of association rules are discovered for classification. One such is CBA (Classification Based Association) method presented in Liu et al. (1998) where association rules are mined for pre determined target class. CBA2 is an extension of CBA where association rules predict different classes having different minimum support to solve data imbalance problem.

A novel classification model based on relational association rule mining has been proposed in Gabriela Czibula et al. (2014) to predict whether a software module is defective or not. The Defect Prediction Relational Association Rules (DPRAR) classifier defines set of relations between the metric values during training process. At the classification stage, a new software entity is declared as positive or negative based on measurement of how ‘close’ the entity is to the positive instances and also how ‘far’ it is from negative one. The computed results have shown that DPRAR classifier performed better than that of the other classifiers – CBA2, 1R and Bagging.
Apart from rule-based methods, several different machine learning algorithms were developed to solve software defect prediction problem. Menzies et al. (2007), evaluated the Naive Bayes classifier, OneR and J48 and concluded that Naive Bayes produced the best results on the 8 used NASA datasets. Another research contribution by Kamruzzaman & Chowdhury Mofizur Rahman (2004) presented text categorization using combination of association rules and Naive Bayes where in instead of using words, word relation is used to derive feature set from pre-classified text documents. Naïve Bayes Classifier is then applied on derived features for final categorization.

Metrics which play key role as feature set were also analysed in the literatures and applied to compare different approaches. Pradeep Singh et al. (2011) evaluated the object-oriented metrics given by Chidamber and Kemerer, and few other static code metrics for couple of open source projects. The empirical study is to find out nature of relationship between these metrics and defects. In another literature study (Danijel Radjenovic et al. 2013), results shows that 49% of the metrics used in defect prediction models are Object-oriented metrics when compared to 27% of traditional source code metrics or 24% of process metrics.

Chidamber and Kemerer’s (CK) object-oriented metrics were most frequently used and these metrics along with process metrics have been successful in finding defects compared to traditional size and complexity metrics. Ludmila Kuncheva (2006) described the value of using McCabe and Halstead static code attributes to learn defect predictor models. Process metrics on the other hand serves better at predicting post-release defects compared to any static code metrics. Lech Madeyski (2014) presents an empirical evaluation in which several process metrics were investigated in
order to identify the ones which significantly improves product metrics based defect prediction models.

2.4 LITERATURE REVIEW ON NEURAL NETWORKS FOR SOFTWARE PREDICTION MODELS

The various works carried out for predicting software defects in a software development process employing variants of neural network models as available in the literature is presented in this section.

Erturk & Sezer (2015) presented the application of the Adaptive Neuro Fuzzy Inference System (ANFIS) for the software fault prediction problem. Bisi & Goyal (2015) proposed an Artificial Neural Network (ANN) model with use of Sensitivity Analysis (SA-ANN) and Principal Component Analysis (PCA-ANN) for dimensionality reduction of the prediction problem. The result shows that non-linear scaling has good effect on predictive capability and PCA-ANN model provides higher accuracy than SA-ANN model and some other existing models for four datasets in early prediction of software fault-prone module detection.

Roy et al. (2014) proposed a robust feed forward neural network (FFNN) based dynamic weighted combination model (PFFNNDWCM) for software reliability prediction.

Malhotra, R. (2014) analyses and compares the statistical and six machine learning methods for fault prediction to find the relationship between the static code metrics and the fault proneness of a module. The result confirms that the predictive capability of the machine learning methods for software fault prediction.
Zaryabi & Ben Hamza (2014) proposed a neural network-based model for optimal software testing and maintenance policy, where the software testing environment and the operational environment are characterized by an environmental factor. Abaei & Selamat (2014) designed a new framework that decreased the effect of irrelevant and inconsistent modules on fault prediction in which the entire project's modules are clustered. Wahono et al. (2014) proposed a combination of genetic algorithm and bagging technique for improving the performance of the defect prediction. The result indicates that the proposed method makes an improvement in neural network prediction performance in software defect prediction.

Wahono et al. (2014) proposed a combination of genetic algorithm and bagging technique for improving the performance of the software defect prediction. Wahono et al. (2014) proposed a comparison framework which aims to benchmark the performance of a wide range of classification models within the field of software defect prediction.

The results show that the proposed predicting model is more efficient than the existing techniques in prediction performance and can be used in software defect prediction. Jin et al. (2012) presented the applications of Artificial Neural Network (ANN) and support vector machine in software fault-prone prediction using metrics.

Hong et al. (2012) presented a novel prediction model using Random Forest classifier for predicting Software fault-proneness. Chatterjee et al. (2012) explored a new approach for predicting software faults by means of NARX neural network. The results computed shows that the proposed approach outperformed the other existing parametric and neural network based software reliability models with a reasonably good predictive accuracy.
Neelamegam & Punithavalli (2012) proposed the use of ensemble classifiers that uses three frequently used classifiers, Back Propagation Neural Network (BPNN), Support Vector Machine (SVM) and K-Nearest Neighbor (KNN). The use of proposed ensemble classifiers concludes that prediction of software module defection can be improved by combining simple and object oriented metrics with multiple classifiers. Wu & Yang (2011) proposed a general regression neural network (GRNN) which makes feasibility without constructing a statistical model like classic software reliability models and having difficulties of solving multivariate likelihood equations. This method can be used to predict software failures.

Catal & Banerjee (2010) demonstrated the procedure for applying paradigm and bio-inspired algorithm for developing software fault prediction models. Zheng (2010) studied three cost-sensitive boosting algorithms to boost neural networks for software defect prediction. The experimental results suggest that threshold-moving is the best choice to build cost-sensitive software defect prediction models with boosted neural networks among the three algorithms studied. Ardi & Sandhu (2010) evaluated a hybrid fuzzy-Genetic Algorithm and Particle Swarm Optimization trained Neural Network techniques empirically. The result shows that the proposed method can be used for the maintenance severity prediction of the software systems.

Goel & Singh (2009) investigated the statistical model, such as logistic regression (LR), and the machine learning approaches, such as Artificial Neural Networks (ANN) for predicting fault proneness. The proposed method can be used in Software fault prediction.

Gray et al. (2009) used the static code metrics for a collection of modules contained within eleven NASA data sets with a Support Vector Machine classifier for software defect prediction. Xu et al. (2008) presented an innovative neural network model that alleviates the various problems
associated with the traditional back-propagation neural network models. The proposed model is not limited to software defect prediction, and can be applied for other estimation problems during software development, such as identifying critical test cases for regression testing.

Yunlong et al. (2008) presented a RBF neural network based on empirical mode decomposition theory for software reliability model prediction.

Zhang & Xu (2005) explored the use of neural networks as a tool for dynamically predicting the number of faults in a program. The result indicates that the neural networks did produce models with better quality of fit and predictive quality than the traditional G-O model for software defect prediction.

2.5 LITERATURE REVIEW ON OPTIMIZATION OF NEURAL NETWORK ARCHITECTURES

Neural network models basically mimic the operation of the human brain; in more specific to be the biological neuron. The developed artificial neural network models gets trained themselves with random initialization of weights during the training process. This random weight initialization results in premature convergence of the network and hence several optimization approaches were used in the early literature to tune the weights of the neural network. This section presents the literature review on optimization of neural network architectures.

Bas et al. (2016) proposed a robust learning algorithm for the multiplicative neuron model ANNs that uses Huber's loss function as fitness function. The training of the multiplicative neuron model is performed using
particle swarm optimization. The result demonstrates that the algorithm is unaffected by outliers and has a robust structure.

Pence et al. (2016) proposed a new unconstrained Global Optimization Method Based on Clustering and Parabolic Approximation (GOBC-PA) to tune the neural network models. Li et al. (2016) proposed the multi-sided multi-granular neural network ensemble optimization method based on feature selection, which divides attribute granularity of dataset from multi-side, and structures multi-granular individual neural networks using different attribute granularity and the corresponding subsets.

Walraven et al. (2016) proposed a new method based on reinforcement learning. The simulation results shows that the resulting the policies are sufficiently robust to deal with inaccurate speed and density measurements. Xu et al. (2016) constructed a hysteretic chaotic neural network with the uncertain term and proposed a novel control strategy based on fuzzy sliding mode to control its dynamic behaviors without changing its structure. Simulation results show that the control strategy is a valid method to control the dynamic behaviors of the neural network and resolve the function optimization problem.

Zhang et al. (2016) constructed the coupled network structure with the Hindmarsh – Rose (HR) neuron cells as the unit and found out the optimal coupling matrix, by using the Chaos Ant Swarm Optimization (CASO) algorithm. The results indicate that proper coupling matrix of HR neural network could be obtained by means of CASO algorithm.

Gundogdu et al. (2016) used Gaussian activation function in multiplicative neuron model artificial neural network. The proposed Gaussian activation function-based multiplicative neuron model artificial neural network has superior forecasting performance when applied to real-life time series. Le

Melo & Watada (2016) proposed Gaussian-PSO-based structural learning and fuzzy reasoning to optimize the weights and the structure of the Feed Forward Neural Network. The proposed method improves the learning and removes the stress by eliminating the necessity of determining a detailed network. Salama & Abdelbar (2015) presented a new algorithm, ANN-Miner, which uses ACO to learn the structure of feed-forward neural networks.

Qi et al. (2015) proposed a hybrid tree-based particle swarm optimization by combining the genetic programming and particle swarm optimization with newly designed genetic operators and evolution rules, algorithm which optimizes the structure and parameters of the neural tree network model simultaneously. The proposed hybrid evolutionary methods can be applied to increase our understanding of the underlying structural and dynamic properties of the brain using only reasonable amounts of behavioral data.

Mirjalili et al. (2015) employed the Grey Wolf Optimizer (GWO) for training Multi-Layer Perceptron (MLP). The statistical result shows that the GWO algorithm is able to provide very competitive results in terms of improved local optima avoidance. Fe et al. (2015) presented a study that shows the need of heterogeneous platform (CPU–GPU–FPGA) to accelerate the optimization of ANNs using genetic algorithms and an implementation of a platform based on embedded systems with hardware accelerators.

Mirjalili et al. (2015) proposed a new trainer using the heuristic algorithm called Social Spider Optimization (SSO) algorithm. The result shows that the proposed FNNSSO is able to provide very promising results compared with other algorithms for non-linear problems.
Maarouf et al. (2015) reviewed the combination of Artificial Neural Networks (ANN) and Evolutionary Optimisation (EO) to solve challenging problems for the academia and the industry.

Dreżewski et al. (2015) proposed a new algorithm for optimization of neural networks architecture based on multi-agent evolutionary approach. Bazoobandi & Eftekhari (2015) proposed a memetic algorithm for tuning Fuzzy Wavelet Neural Network (FWNN) parameters in an adaptive way which combines Particle Swarm Optimization (PSO) as an evolutionary algorithm and an innovative local search which is based on a Fuzzy Inference System (FIS).

Cao et al. (2015) proposed an improved BSO algorithm with differential evolution strategy and new step size method. The comparative experimental results illustrate that the proposed algorithm performs significantly better than the original BSO. Prasad et al. (2015) proposed a hybridization of Particle swarm optimization and Genetic algorithm with the back propagation learning based Multilayer perceptron neural network.

2.6 OUTCOME OF THE LITERATURE REVIEW AND MOTIVATION FOR THE RESEARCH WORK

Section 2.2 through section 2.5 has presented the earlier literature works carried on various software defect prediction models, variant neural network architectures developed for software fault prediction, optimization of neural network algorithms and progress made in association rule mining and Naives Bayes approach.

From the literature reviews made, it is noted that the numerous methods proposed for carrying out the software defect analysis and for its prediction has achieved solutions. But each of the methods possesses their
own drawbacks resulting in delayed convergence or premature convergence without reaching the required solution point or identifying the parts which are not defective in nature. The earlier proposed methodologies are found to question the reliability of the predictive models and it is not guaranteed. Numerous algorithmic defect predictive models tend to get trapped in the local and global minimal point without reaching the near solution point. Rule mining is noted to elapse more computational time and levy heavily on the computational burden of the model. Since, the existence of all these traps occur at any one point of the prediction process flow, the reliability of the predictor model is questioned.

To overcome all the above said drawbacks observed from the study made on early literatures, this thesis work concentrates on developing scalable and reliable software defect predictor models by achieving the learning and generalization ability. The thesis is focussed to develop defect predictor models that facilitate the software fault prediction process with complete guarantee on the specified given constraints in an effective way for the considered software datasets.

2.7 SUMMARY

In this thesis chapter, a detailed literature review on the existing approaches employed to carry out software defect prediction during the software development process is presented. The drawbacks of the existing defect predictive models are clearly brought out and this authenticates the motivation for this research work. The extensive literature review has given an insight on the existing software defect predictive models, numerous neural network architectures employed for prediction, several neural network optimization approaches and the various association rule mining methods.