International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 - 5161

Available at http://inpressco.com/category/ijcet

Research Article

A Hybrid Model of Soft Computing Technique for Software Fault Prediction

Anurag Shrivastava[Å*] and Vishal Shrivastava[Å]

[Å]ACE&IT Jaipur, India

Accepted 16 July 2014, Available online 01 Aug 2014, **Vol.4, No.4 (Aug 2014)**

Abstract

Software fault prediction plays a vital role in software quality assurance, identifying the faulty modules to better concentrate on those modules and helps to improve the quality of the software. With increasing complexity of software now a day's feature selection is important to remove the redundant, irrelevant and erroneous data from the dataset. In general, Feature selection is done mainly based on filter and wrapper .In this paper wrapper method for feature selection is used, and various machine learning techniques (Neural gas, SVM classifier), Symbolic Regression(genetic programming) and ABC algorithm are used. Artificial Bee Colony algorithm is considered new and widely used in searching for optimum solutions. ABC proved to be a suitable candidate for classification tasks, which gives a better prediction than the traditional methods. NASA's public dataset KC1 and PC1 available at promise software engineering repository is used. And also MUSHROOM dataset taken from the Audubon Society Field to evaluate the performance of the software fault prediction models Accuracy value are used. Software development has become an essential investment for many organizations. Software engineering practitioners have become more and more concerned about accurately predicting the fault and quality of software under development. Accurate estimates are desired but no model has proved to be successful at effectively and consistently predicting software fault.

Keywords: *Neural Gas (NG), Support Vector Machine (SVM), Genetic Programming (GP), Artificial Bee Colony (ABC).*

1. Introduction

1.1 Software Fault Prediction

Due to the increased dependency of modern system on software-based system, software reliability and quality has become the primary concern during the software development. It is difficult to produce fault-free software due to the problem complexity, complexity of human behaviors, and the resource constrains. System failures due to the software failure are common and results in undesirable consequences, which can adversely affect both reliability and safety of the system. A Software System consists of various modules and in general

It is known that a small number of software modules are responsible for majority of the failures. Moreover, it is also known that early identification of faulty module can help in producing software of quality and reliability more cost effectively. Therefore, it is desirable to classify the software module as fault-prone (FP) or not fault-prone (NFP) just after the coding phase of software development. Once the modules are classified as FP or NFP, more testing efforts can be put on the fault-prone module to produce reliable software.

A fault is a defect in source code that causes failures when executed. Software modules are said to be fault-prone, when there are a high probability of finding faults during its operation. In other words, a fault-prone software module is the one containing more number of expected faults than a given threshold value. The threshold value can take any positive value and depends on the project specific criteria. In general, when a software module has been identified as fault-prone, more and more testing efforts are applied to improve its quality and reliability. The amount of testing effort applied to a faulty module should be proportional to its degree of fault-proneness. In other words, it is undesirable to apply equal amount of testing efforts to all the software modules. The testing efforts should be allocated on the basis of its degree of fault-proneness (Ajeet Kumar Pandey *et al*, 2010).

Fault prediction model has to consider two main things namely, accuracy and complexity. Better accuracy and least complexity can be the aim of developing a fault prediction model. To predict the faults techniques like statistical methods, machine learning methods, parametric models and mixed algorithms are used. Supervised machine learning technique is a method in which input and output in hand is used to train the machine to better predict in future. KC1, PC1 and mushroom dataset available at promise repository is used for the experiments.

This paper proposes a model for prediction of fault-prone software module using machine learning, Genetic Programming and ABC classifier.

1.2 Wrapper feature selection

Feature selection is a pre- processing step in which irrelevant,

*Corresponding author **Anurag Shrivastava** is a M.Tech Scholar and **Vishal Shrivastava** is working as Associate Professor

Redundant, missing and erroneous data are removed. Feature selection plays a vital role in software fault prediction as complex software's have thousands and thousands of lines of code and has lot of attribute value pairs. Filters and wrappers are the two categories of feature selection. Wrapper based feature selection are computationally complex as they are based on complex classification algorithms. Filters are simple as they are based on the characteristics of data and hence preferred generally for large dataset. In this proposed model optimum feature selection is done by wrapper from the filtered features (C.Akalya devi1 *et al*, 2012).

I predict fault proneness model using genetic programming and machine learning methods. One of the approaches to identify faulty classes early in the development cycle is to predict models by using software metrics.

In order to achieve this aim I have used dataset collected from NASA PROMISE Software Engineering Repository data set and also MUSHROOM dataset taken from The Audubon Society Field. The different dataset used by me will provide an important insight to researchers for identifying the relevance of metrics with a given type of dataset.

2. Literature Review

Significant work has been done in the field of fault detection. Highlights of select papers have been discussed in this section. There are various categories of methods to predict faulty classes such as machine learning methods, statistical methods, etc. We have observed that much of the previous work used traditional statistical methods to bring out the results, but very few studies have used machine learning methods. Recently, the trend is shifting from traditional statistical methods to modern machine learning methods. The most common statistical methods used are univariate and multivariate logistic regression. A few key points of the papers using machine learning and statistical methods are discussed.

Various efforts have been made for software fault prediction using methods such as Decision Trees, Neural Networks, Support Vector Machines, Bayesian Methods, Naïve Bayes, Fuzzy Logic, Dempster–Shafer Belief Networks, Genetic Programming, Case based Reasoning, Logistic Regression and ABC algorithm.

On reviewing literature, it is found that various machine learning approaches such as supervised, semi-supervised, and unsupervised have been used for building a fault prediction models. Among these, supervised learning approach is widely used and found to be more useful for predicting fault-prone modules if sufficient amount of fault data from previous releases are available. Generally, these models use software metrics of earlier software releases and fault data collected during testing phase. The supervised learning approaches cannot build powerful models when data is limited. Therefore, the some researchers presented a semi-supervised classification approach for software fault prediction with limited fault data.

(Menzies *et al*, 2007) shown that defect predictors can be learned from static code attributes since they are useful, easy to use, and widely used. Taking clues from Pandey and Goyal (C. Catal *et al*, 2009) have presented an early fault prediction model using process maturity and software metrics. Software metrics have been shown to be good indicators of software quality, and the number of faults present in the software. Software metrics, which represent the software product and its process, can be collected relatively earlier during software development.

(Catal and Diri et, al 2009) investigated the effects of data size, metrics, and the feature selection techniques for software fault prediction. They have also presented a systematic review of various software fault prediction studies (T. Menzies *et al*, 2007). From the literature, it has been found that the decision tree induction algorithms such as CART, ID3 and C4.5 are efficient technique for module classification. These algorithms uses crisp value of software metrics and classify the module as a fault-prone or not fault-prone. It has been found that most of the early phase software metrics have fuzziness in nature and crisp value assignment seems to be impractical. Also a software module can't be completely fault-prone or not fault prone.

In other words it is unfair to assign a crisp value of software module representing its fault proneness.

A few works has also done in feature selection, removing irrelevant and redundant features from the dataset, feature selection helps improve the performance of learning models by (C. Akalya devi1 et, al 2012)

- Lessening the effect of the curse of dimensionality.
- Improving generality capability.
- Fast up learning process.
- Improving model interpretability.

Feature Selection Algorithms selects a set of 'm' features from set of 'n' features, Where m <n.Filters are simple hence preferred for large dataset but less accurate than computationally complex wrappers.

In (Parvinder Singh Sandhu et, al 2007) various classifiers like Naïve Bayes, Logistic Models Trees, RBF Network, and Classification via Regression, RBF Network, and Simple Logistic are used for KC1 dataset with 2107 modules. However, no feature selection algorithm was used. Thus, using all features will increase classifier complexity. Max accuracy obtained is 70.99% only for CvR classification.

Naive Bayes gave a least accuracy value of 62.46% only. In (N.Gayatri *et al*, 2010) proposed a feature selection method namely Decision Tree Induction Rule based (DTIRB) feature set. DTIRB uses three decision tree algorithms J48, CART, BFTree for feature selection. For KC1 dataset 15 features where selected by DTIRB. This model is evaluated with 18 classifiers and Naive Bayes showed a better result of 0.25 MAE and 0.48 RMSE only.

2.1 Methodology

I proposed framework consists of two parts: scheme evaluation and defect prediction. The scheme evaluation focuses on evaluating the performance of a learning scheme, while the defect prediction focuses on building a final predictor using historical data according to the learning scheme and after which the predictor is used to

predict the defect-prone components of a new (or unseen) software system.

A learning scheme is comprised of:

1. A data preprocessor,
2. An attribute selector,
3. A learning algorithm.

3. Proposed Software Defect Prediction Framework

3.1 Overview of the Framework

Generally, before building defect prediction model and using them for prediction purposes, we first need to decide which learning scheme should be used to construct the model. Thus, the predictive performance of the learning scheme should be determined, especially for future data. However, this step is often neglected and so the resultant prediction model may not be trustworthy. Consequently, we propose a new software defect prediction framework that provides guidance to address these potential shortcomings.

The framework consists of two components

1) Scheme evaluation
2) Defect prediction

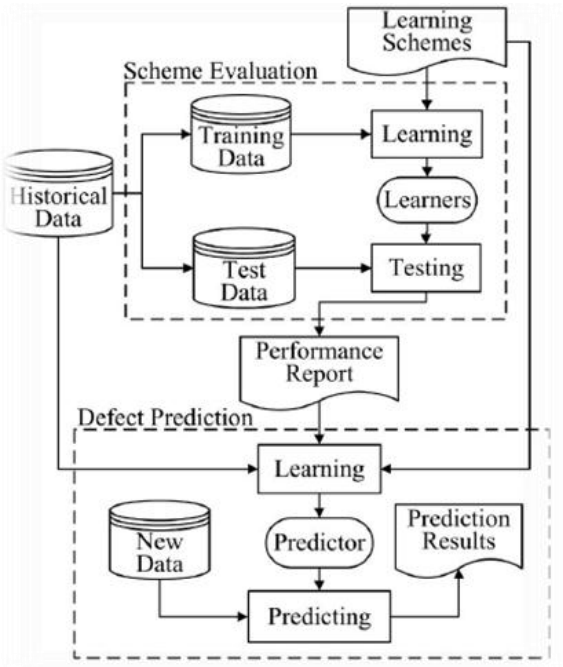

Fig 3.1 Scheme Evaluation

From Fig. 3.1, we can see that the historical data are divided into two parts: a training set for building learners with the given learning schemes, and a test set for evaluating the performances of the learners. It is very important that the test data are not used in any way to build the learners. This is a necessary condition to assess the generalization ability of a learner that is built according to a learning scheme and to further determine whether or not to apply the learning scheme or select one best scheme from the given schemes.

At the defect prediction stage, according to the performance report of the first stage, a learning scheme is selected and used to build a prediction model and predict software defect. Fig3. 1, we observe that all of the historical data are used to build the predictor here. This is very different from the first stage; it is very useful for improving the generalization ability of the predictor. After the predictor is built, it can be used to predict the defect-proneness of new software components

3.2 Scheme Evaluation

The scheme evaluation is a fundamental part of the software defect prediction framework. At this stage, different learning schemes are evaluated by building and evaluating learners with them. Fig. 3.2 contains the details. The first problem of scheme evaluation is how to divide historical data into training and test data. As mentioned above, the test data should be independent of the learner construction. This is a necessary precondition to evaluate the performance of a learner for new data. Cross-validation is usually used to estimate how accurately a predictive model will perform in practice. One round of cross validation involves partitioning a data set into complementary subsets, performing the analysis on one subset, and validating the analysis on the other subset. To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds

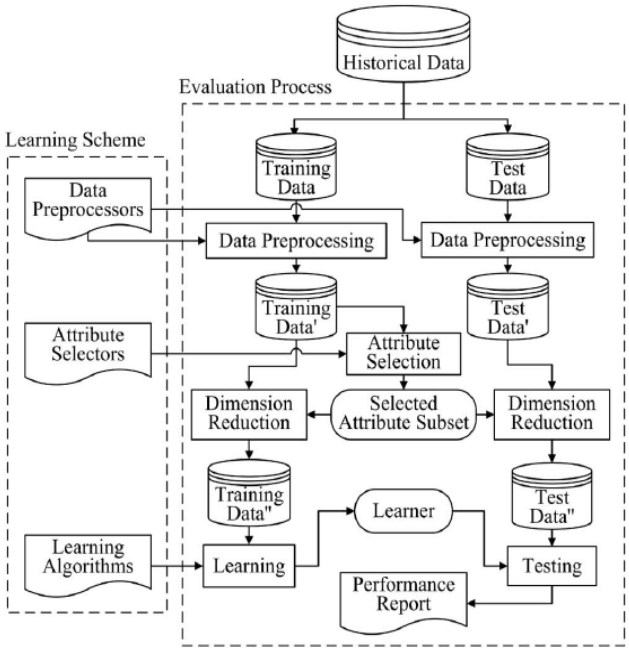

Fig 3.2 Evaluation Process

In my framework, an M_ N-way cross-validation is used for estimating the performance of each predictive model, that is, each data set is first divided into N bins, and after that a predictor is learned on (N-1) bins, and then tested on the remaining bin. This is repeated for the N folds so that

each bin is used for training and testing while minimizing the sampling bias. To overcome any ordering effect and to achieve reliable statistics, each holdout experiment is also repeated M times and in each repetition the data sets are randomized. So overall, M_ N models are built in all during the period of evaluation; thus M_ N results are obtained on each data set about the performance of the each learning scheme.

After the training-test splitting is done each round, both the training data and learning scheme(s) are used to build a learner. A learning scheme consists of a data preprocessing method, an attribute selection method, and a learning algorithm. The detailed learner construction procedure is as follows:

1. Data preprocessing. This is an important part of building a practical learner. In this step, the training data are preprocessed, such as removing outliers, handling missing values, and discrediting or transforming numeric attributes. In our experiment, we use a log-filtering preprocessor which replaces all numeric n with their logarithms ln(n), such as used in MGF.
2. Attribute selection. The data sets may not have originally been intended for defect prediction; thus, even if all of the attributes are useful for its original task, not all may be helpful for defect prediction. Therefore, attribute selection has to be performed on the training data. Attribute selection methods can be categorized as either filters or wrappers [3]. It should be noted that both filter and wrapper methods only operate on the training data. A filter uses general characteristics of the data to evaluate attributes and operates independently of any learning algorithm. In contrast, a wrapper method exists as a wrapper around the learning algorithm searching for a good subset using the learning Algorithm itself as part of the function evaluating attributes subsets. Wrappers generally give better results than filters but are more computationally intensive. In our proposed framework, the wrapper attribute selection method is employed. To make the most use of the data, we use an M_ N-way cross-validation to evaluate the performance of different attribute subsets.
3. Learner construction: Once attribute selection is finished, the preprocessed training data are reduced to the best attribute subset. Then, the reduced training data and the learning algorithm are used to build the learner. Before the learner is tested, the original test data are preprocessed in the same way and the dimensionality is reduced to the same best subset of attributes. After comparing the predicted value and the actual value of the test data, the performance of one pass of validation is obtained. As mentioned previously, the final evaluation performance can be obtained as the mean and variance values across the M_ N passes of such validation.

3.3 Learning scheme

N fold cross validation most common 10-fold cross validation method is used for detecting faulty instances. One round of cross validation is partitioning a sample of data into complementary subsets, performing the analysis on one set known as training set and validating the analysis on other set known as testing set. In 10-fold one to 9 of them is treated as training set and one as testing set. This is repeated 10 times where each one of the subset is used as training set once.

A cross-validation approach gives a more realistic accuracy assessment. It involves dividing the whole data set in to multiple training and test sets. The estimation results in the training stages present the estimation accuracies from the model construction data sets. The test stage evaluates the estimation accuracies of the models using the other unseen data sets. In the training stage, a cross-validation approach calculates the accuracy for each training data set, and then combines the accuracies across all training result. In the test stage, it calculates the accuracy for each test set, and then combines the accuracies a cross all test sets for the test result. We perform 10-fold cross-validation.

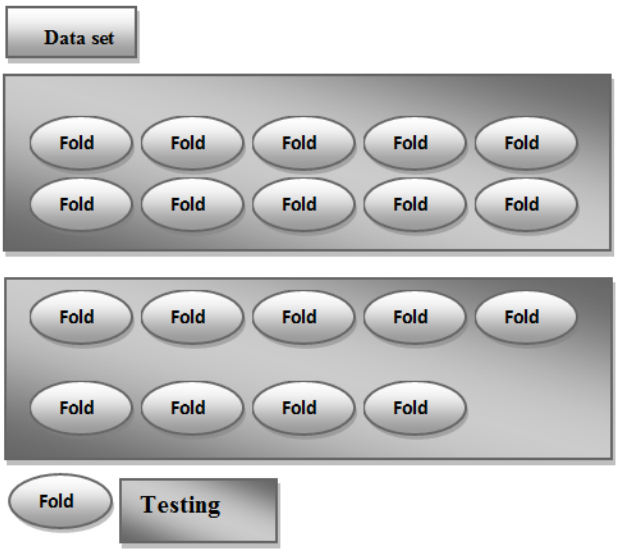

Fig: 3.3 N Fold cross Validation

3.4 Defect Prediction

The defect prediction part of my framework is straightforward; it consists of predictor construction and defect prediction. During the period of the predictor construction

1. A learning scheme is chosen according to the Performance Report.
2. A predictor is built with the selected learning scheme and the whole historical data. While evaluating a learning scheme, a learner is built with the training data and tested on the test data. Its final performance is the mean over all rounds. This reveals that the evaluation indeed covers all the data. However, a single round of cross-validation uses only one part of the data. Therefore, as we use all of the historical data to build the predictor, it is expected that the constructed predictor has stronger generalization ability.
3. After the predictor is built, new data are preprocessed in same way as historical data, then the Constructed predictor

Table 3.1 Some main characteristics of these datasets

	Project	Attributes	Language	No of modules	%of defective module	Faulty Module no.	Description
Procedural	PC1	43	C	1107	6.9	77	Flight Software for an earth orbiting satellite.
Object Oriented	KC1	27	C++	2107	15.4	326	Storage management for receiving and processing ground data collection, Processing and delivery of satellite metadata.

can be used to predict software defect with preprocessed new data. The detailed defect prediction process is described with pseudo code in the following Procedure Prediction.

3.5 Dataset

In my paper I have used three kind of data set, one is pc1, second is kc1 and third last one mushroom apply machine learning techniques, genetic programming, and ABC algorithm on the given dataset and calculate accuracy. The detail description of first dataset given below:-

The datasets used in this thesis are four mission critical NASA software projects, which are publicly accessible from the repository of the NASA IV&V Facility Metrics Data Program. One dataset PC1 is from software projects written in a procedural language (C) where a module in this case is a function. The other dataset KC1 are from projects written in object-oriented languages (C++ and Java) where a module in this case is a method. Each dataset contains 21 software metrics (independent variables) at the module-level and the associated dependent Boolean variable: Defective (whether or not the module has any defects) (Karim O. Elish *et al*, 2008)

And last dataset Mushroom records drew from The Audubon Society Field Guide to North American Mushrooms (1981). This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family (pp. 500-525). Each species is identified a definitely edible, definitely poisonous, or of unknown edibility and not recommended.

3.6 Prediction performance measures

The performance of prediction models for two-class problem (e.g. defective or not defective) is typically evaluated using a confusion matrix, which is shown in Table 3.4.

In this study, we used the commonly used prediction performance measures accuracy to evaluate and compare prediction models quantitatively. These measures are derived from the confusion matrix.

3.6.1. Accuracy

Accuracy is also known as correct classification rate. It is defined as the ratio of the number of modules correctly predicted to the total number of modules. It is calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Table 3.2 A Confusion Matrix

		Predicted	
		Not defective	Defective
Actual	Not defective	TN = True Negative	FP = False Positive
	Defective	FN = False Negative	TP = True Positive

3.7 Feature Selection

Feature selection is important to remove the redundant, Irrelevant and erroneous data from the dataset. In general, Feature selection is done mainly based on filter and wrapper.

Feature selection is a process that selects a subset of original features. Feature selection is one of the important and frequently used techniques in data preprocessing for data mining. In real-world situations, relevant features are often unknown a priori. Hence feature selection is a must to identify and remove are irrelevant/redundant features. It can be applied in both unsupervised and supervised learning.

3.7.1 The Wrapper Approach for Feature Selection

Wrapper model approach uses the method of classification itself to measure the importance of features set; hence the feature selected depends on the classifier model used. Wrapper methods generally result in better performance than filter methods because the feature selection process is optimized for the classification algorithm to be used. However, wrapper methods are too expensive for large dimensional database in terms of computational complexity and time since each feature set considered must be evaluated with the classifier algorithm used.

3.8 Use of Various Machine Learning Methods:-

After feature selection, refer the data to make cluster using neural gas algorithm the input to the algorithm is a data set and the desired output is accuracy.

3.8.1 A Neural Gas Approach for Finding Fault Proneness of Modules

There are numbers of software quality prediction models based upon genetic algorithms, artificial neural

network and other data mining algorithms. One of the promising aspects for quality prediction is based on clustering techniques. Most quality prediction models that are based on clustering techniques make use of K-means, Mixture-of-Gaussians, Self-Organizing Map, Neural Gas and fuzzy K-means algorithm for prediction. In all these techniques a predefined structure is required that is number of neurons or clusters should be known before we start clustering process. But in case of Neural Gas there is no need of predetermining the quantity of neurons and the topology of the structure to be used and it starts with a minimal neurons structure that is incremented during training until it reaches a maximum number user defined limits for clusters. Hence, in this work we have used Neural Gas as underlying cluster algorithm that produces the initial set of labeled cluster from training data set and thereafter this set of clusters is used to predict the quality of test data set of software modules. The best testing results shows 80% accuracy in evaluating the quality of software modules. Hence, the proposed technique can be used by programmers in evaluating the quality of modules during software development (Saravjeet Kaur *et al*, 2010)

3.8.1.2 Calculate accuracy from supervised learning using SVM classifier.

Support Vector Machines (SVMs) provide a powerful method for classification (supervised learning). Since the mid of 1990, SVM appeared with the continuous development and maturation of machine learning theory. It has exhibited performance which superior to the other existing methods. SVM is a novel learning machine introduced first by Vapnik (V.Vapnik *et al*, 1995). It is a theory of machine learning focusing on small sample data based on the structural risk minimization principle from computational learning theory. Hearst *et al*. (M.A Hearst *et al*, 1998) positioned the SVM algorithm at the intersection of learning theory and practice: ''it contains a large class of neural nets, radial basis function (RBF) nets, and polynomial classifiers as special cases. Yet it is simple enough to be analyzed mathematically, because it can be shown to correspond to a linear method in a high dimensional feature space nonlinearly related to input space.'' In this sense, support vector machines can be a good candidate for combining the strengths of more theory-driven and easy to be analyzed conventional statistical methods and more data-driven, distribution free and robust machine learning methods.

3.8.2 Symbolic Regression

Symbolic regression, an application domain of genetic programming (GP), aims to find a function whose output has some desired property, like matching target values of a particular data set. While typical regression involves finding the coefficients of a pre-defined function, symbolic regression finds a general function, with coefficients, fitting the given set of data points. The concepts of symbolic regression using genetic programming can be used to evolve a model for faulty module predictions. Such a model has the advantages that the evolution is not dependent on a particular structure of the model and is also independent of any assumptions, which are common in traditional time-domain parametric software reliability growth models. This research aims at applying experiments targeting fault predictions using genetic programming and comparing the results with traditional approaches to compare efficiency gains.

Under this scenario, what becomes significantly interesting is to have modeling mechanisms that can exclude the pre-suppositions about the model and is based entirely on the fault data. This is where the application of symbolic regression using genetic programming (GP) becomes feasible.

3.8.2.2 GP design

The representation of solutions in the search space is a symbolic expression in the form of a parse tree, which is a structure having functions and terminals.
The quality of solutions is measured using an evaluation function. A natural evaluation measure for symbolic regression problems is the calculation of the difference between the obtained and expected results in all fitness cases, åni =1 | ei −e0i | where ei is the actual fault count data, e0
i is the estimated value of the fault count data and n is the size of the data set used to train the GP models.

Various variation operators can be used to grow or shrink a variable length parse tree. Similarly, there are various selection mechanisms that can be used to determine individuals in the next generation. The effectiveness of these operators is problem-dependent. In our experiments, we have used cross-over with branch swapping by randomly selecting nodes of the two parent's trees. We have also used mutation in which a random node from the parent tree is substituted with a new random tree created with the available terminals and functions. A small proportion of individuals were also copied into the next generation without any action of operators. The selection mechanism selected a random number of individuals from the population and chose the best of them; if two individuals were equally fit, the one having the less number of nodes was chosen as the best.

3.8.2.3 Symbolic regression feature

- Symbolic regression demos
- Graphical display of results of symbolic regression.
- Statistical analysis of multigene symbolic regression models†.

3.8.2.4 I have also used ABC(Artificial Bee Colony) based Data Mining Algorithms for Classification Tasks.

Artificial Bee Colony (ABC) algorithm is considered new and widely used in searching for optimum solutions. This is due to its uniqueness in problem-solving method where the solution for a problem emerges from intelligent behavior of honeybee swarms. I use the ABC algorithm as a new tool for Data Mining particularly in classification

tasks. Moreover, the proposed ABC for Data Mining were implemented and tested against six traditional classification algorithms classifiers. From the obtained results, ABC proved to be a suitable candidate for classification tasks. This can be proved in the experimental result where the performance of the proposed ABC algorithm has been tested by doing the experiments using datasets. The results obtained in these experiments indicate that ABC algorithm is competitive, not only with other evolutionary techniques, but also to industry standard algorithms such as PART, SOM, Naive Bayes, Classification Tree and Nearest Neighbor (KNN), and can be successfully applied to more demanding problem domains.

3.9 Experimental Design and Result

For measuring the performance of the proposed ABC data mining algorithm, we have conducted experiment with different datasets that are selected from the NASA Repository. I have tested on the three popular data sets in the experiment. To show the performance of ABC classification algorithm, we have compared it with three other classification algorithms.

4. Experiment and Results

In this chapter we present the results of proposed algorithm when run on different datasets. This chapter also has some sub sections to describe the outcomes in step by step manner. In first section we describe the selection process of datasets and description of that dataset. After that we present the results when run on NASA PC1 Dataset and then on KC1 dataset and then apply on Mushroom dataset.

4.1 Dataset

The datasets used in this thesis are four mission critical NASA software projects, which are publicly accessible from the repository of the NASA IV&V Facility Metrics Data Program. One dataset PC1 is from software projects written in a procedural language (C) where a module in this case is a function. The other dataset KC1 are from projects written in object-oriented languages (C++ and Java) where a module in this case is a method. Each dataset contains 21 software metrics (independent variables) at the module-level and the associated dependent Boolean variable: Defective (whether or not the module has any defects). And last dataset Mushroom Records had drawn from The Audubon Society Field Guide to North American Mushrooms (1981).

4.2 Result Analysis

I implement my proposed hybrid technique using MATLAB7.14 suite is used for analyzing the prediction accuracy To predict the software fault our proposed mechanism rule we first find the various machine learning technique, Genetic Programming result, then after I apply ABC technique.

I have used different methods to build more than one classifier and now wish to compare accuracy .But what is accuracy? How we can be estimating it?

I use these measures in technique for accuracy estimation 10-fold cross validation method.

Classifier accuracy measures

Using training data to drive a classifier or predictor and then to estimate the accuracy of the resulting learned model. Accuracy is better measured on a test set consisting of class-labeled tuples that were not used to train the model. The accuracy of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier.

Table 4.2 Prediction performance measures on PC1 dataset using machine learning techniques

Name of Technique (Classifier's)	**Parameter**	**Result**	**Previous Result's**
Neural Gas	Accuracy	94.20	93.30
SVM	Accuracy	95.78	93.10
ABC	Accuracy	91.70	-
Symbolic Regression	Accuracy	87.50	-

The confusion matrix is a useful tool for analyzing how well your classifier can recognize tuples of different classes.

I apply these techniques (NG, SVM, ABC, and Symbolic Regression) on PC1 dataset, and compare with previous results and found that my hybrid model predict more accurate result than previous model's, I use various classifiers but SVM classifier predicts more accurate result on PC1 dataset. Result is shown in Table 4.2.

I apply these techniques (NG, SVM, ABC, and Symbolic Regression) on KC1 dataset, and compare with previous

Results and found that my hybrid model predict more accurate result than previous models, I use various classifiers but SVM classifier predicts more accurate result on KC1 dataset. Result is shown in Table 4.3.

Table 4.3 Prediction performance measures on KC1 dataset using machine learning technique

Name of Technique (Classifier's)	**Parameter**	**Result**	**Previous Result's**
Neural Gas	Accuracy	94.10	Work not done
SVM	Accuracy	94.12	84.59
ABC	Accuracy	92.20	74.07
Symbolic Regression	Accuracy	89.50	Work not done

Here I apply various classifiers on Mushroom dataset, till now no one use mushroom dataset, in mushroom dataset ABC classifier predicts more accurate result. Result is shown in Table 4.4.

Table 4.4 Prediction performance measures on mushroom dataset using machine learning.

Name of Technique (Classifier's)	Parameter	Result	Previous Result's
Neural Gas	Accuracy	94.25	Work not done
SVM	Accuracy	93.21	Work not done
ABC	Accuracy	95.81	Work not done
Symbolic Regression	Accuracy	91.26	Work not done

Conclusion

Comparison of results shows that hybrid model gives better performance. Reduced feature set plays an important role in software fault prediction. This reduced feature set (using wrapper feature selection) improves the performance of learning algorithm and also reduces the computational cost. This further reduces the complexity of the classifier also. From the results the hybrid model with SVM and ABC better predicts the faults. Hence accuracy is better and also the complexity of the classifier is reduced with reduced feature set that is selected by the hybrid model approach. This prediction model will help to pay more attention to fault prone modules in future versions of same software or similar products and also will improve the productivity, easy maintenance and also the quality of software.

Future works

One direction of future work would be conducting additional empirical studies with other datasets to further support the findings of this paper, and to realize the full potential and possible limitation of SVM. Another possible direction of future work would be considering additional independent variables if information on such metrics is available. Finally, it would be interesting to apply hybrid techniques in predicting other software quality attributes in addition to defects.

Reference

Ajeet Kumar Pandey (2010), Predicting Fault-prone Software Module Using Data Mining Technique and Fuzzy Logic, *International Journal of Communication and Computer Technologies (IJCCTS), Vol. 2 Issue 2, 3, 4.*

B.Surendiran3, M.E (CSE), Sri Shakti (2012), Hybrid feature selection model for software fault prediction, *International Journal on Computational Sciences & Applications (IJCSA), Vo2*, and No.2.

R.Kohavi (1997), Wrapper for feature selection for machine learning, *Science direct*, vol. 97,pp-273-324,

T. Kenzie's, J. Greenwald and A. Frank (2007), Data Mining Static Code Attributes to Learn Defect Predictors, *IEEE Transactions on Software Engineering*, vol. 33, pp. 2-13.

C. Catal and B. Diri (2009), Investigating The Effect Of Dataset Size, Metrics Set, and Feature Selection Techniques on Software Fault Prediction Problem, *Information Sciences*, vol. 179, pp. 1040-1058.

C. Catal and B. Diri (2009), A Systematic Review of Software Fault Predictions studies, *Elsevier*, vol. 36, pp. 7346- 7354.

T. Menzies, J. Greenwald, and A. Frank (2007), Data Mining Static Code Attributes to Learn Defect Predictors, *IEEE Trans. Software Eng.*, vol. 33, no. 1, pp. 2-13

Parvinder Singh Sandhu, Sunil Kumar and Hardeep Singh, (2007), Intelligence System for Software Maintenance Severity Prediction, *Journal of Computer Science Publications* 3 (5): 281- 288, 2007, ISSN 1549-3636.

N.Gayatri, Nickolas, A.V.Reddy, (2010), Feature Selection Using Decision Tree Induction in Class level Metrics Data Set for Software Defect Predictions, World *Congress on Engineering and Computer Science* 2010 Vol I.

Karim O. Elish, Mahmoud O. Elish, (2008), Predicting Defect- prone software modules using Support vector Machines, *Journal of Science and software*, Vol 81.

Saravjeet Kaur, Nisha Ratti, Dr. Parvinder S. Sandhu (2010), A Natural Neural Gas Approach for Finding Fault Proneness of Modules in Open Source Software, *International Journal of Research in Engineering and Technology (IJRET)* Vol. 1

V.Vapnik (1995), Nature of Statistical Learning Theory. New York, Springer-*Verlag*, ISBN: 0-387-94559-8.

M.A Hearst, S.T. Dumais, E.Osman, J. Platt, B.Scholkopf (1998), Support Vector Machines, *IEEE Intelligent*