IJARCCE

ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 7, July 2021

DOI 10.17148/IJARCCE.2021.10783

Software Fault-proneness Prediction using Random Forest

Dr.Indumathi SK[1], M.Bharath[2]

[1]Professor of Dr Ambedkar Institute of Technology, Dept of MCA, Bangalore-560056, Karnataka, India

[2]Student of Dr Ambedkar Institute of Technology, Dept of MCA, Bangalore-560056, Karnataka, India

Abstract: Many metric-based classification models have been developed and applied to software fault-proneness prediction. This paper presents a novel prediction model using Random Forest classifier. Random Forest (RF) can be a promising candidate for software quality prediction because it is one of the most accurate classification algorithms available and has strengths in noise handling and efficient running on large data sets. The RF model is constructed and the attribute selection process of the input data is performed before the model evaluation. Prediction accuracy of the model is evaluated using two prediction error measures, Type I and Type II error rates, and compared with well-known prediction models, MultiLayer Perceptron (MLP) neural network model and Support Vector Machine (SVM) model. The results show that the RF model significantly outperforms the SVM model and slightly outperforms the MLP model.

Keywords: fault-proneness, prediction model, random forest

1. INTRODUCTION

As software engineering technology is getting to be advanced, fault-proneness prediction of software modules at early phases becomes more and more important because the problems in early phases largely affect the quality of the late products. In general, it is known that a small number of software modules are responsible for the vast majority of all the faults [1]. This means that early detection of these modules and quality monitoring provide successful project management through the accurate planning of resource allocation and improving the effectiveness of the testing process. Software fault-proneness cannot be directly measured, but it can be estimated based on software complexity metrics [2]. Nevertheless, estimating the exact number of faults is difficult and often not necessary, especially in the early stages when too little information is available [3]. Therefore, in our study, a software fault-proneness prediction model means a software classification model that classifies the design entities into fault-prone and non fault-prone categories.

Many software classification models using complexity metrics as their input vectors have been suggested since 1990s until now. Most of them use supervised algorithms using enough training data, which are based on machine learning and statistical methods such as neural networks, genetic programming, case -based reasoning, decision trees, Bayesian models, fuzzy classification, support vector machines, discriminant analysis, and logistic regression[4, 5]. However, very few organizations have their own training data to build their prediction models. In these cases when training data are not or partly available, new models using unsupervised or semi-supervised algorithms can be used [5]. [6] and [7] focused on Random Forest using the public NASA data sets [8] to compare its prediction performance with other methods. Their experimental results showed that Random Forest generally achieved better performance than compared methods especially in large data sets. The objective of this study is to evaluate the capability of Random Forest (RF) in predicting fault-proneness of software design entities and compare its prediction performanc against two well -known machine learning models.

2. RANDOM FOREST MODEL

Ensemble methods are classification techniques for improving classification accuracy by aggregating the predictions of multiple classifiers. Random Forest is an ensemble classifier that manipulates its input features and uses decision trees as its base classifiers [10]. The process of construction and use of Random Forest, like other ensemble methods, has three steps. Step 1: Create multiple data sets from the original training data. A random subset of input variables, random vector, is chosen and each training set is formed by randomly choosing samples with replacement, bootstrap sampling, from the original data. About one- third of the data are left out of the bootstrap sample and not used in the construction of a decision tree. Randomization helps to improve the generalization error of the ensemble by reducing the correlation among decision trees. Step 2: Build multiple decision trees with the sampled data sets. A random vector is used for tree growing process. For each node, the best split is calculated using input variables in a random vector to decide to split a node. The tree is fully grown without any pruning. Step 3: Combine the decision trees. After all the trees are constructed, the output value is obtained by combining the predictions using a majority voting scheme.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 7, July 2021

DOI 10.17148/IJARCCE.2021.10783

In our experimental study, we use the open source machine learning tool, WEKA [11], to construct a RF model. The number of trees to be generated is set to 10 and the number of input variables in a random vector is set to 3. The data left out of the sample in step 1, out-of- bag (oob) data, can be used as a test data set for classification to get an unbiased estimate of the test set error, oob error estimate. Though this error estimate can be a measure to evaluate prediction performance of a RF model, we use separate test data sets for comparison with other prediction models.

3. EXPERIMENT

3.1 Data Set

The data sets used in this paper are from our earlier study [9]. The experiments are performed only using the metrics for architectural design phase suggested in [12] to quantify SDL(Specification and Description Language) blocks. A block is quantified to a metric vector that has a form of (BRS, EBS, EBC, BP, BS, BR). The elements of a metric vector are independent variables of a prediction model. The output of a model is a FP metric that determines the fault-proneness of a SDL block. FP is 1 when a block is fault-prone, otherwise it is 0. Two training data sets and two test data sets are used for our experimental study. Table
1 summarizes main characteristics of the data sets. train2 and test2 should not contain ambiguous blocks which are difficult to be decided whether they are fault-prone or not, while train1 and test1 may have ambiguous blocks. All data sets are prepared in the form of two types: normalized and raw(not-normalized) data. Experimental results show that the oob error estimates of train1 and train2 are 0.12 and 0.08 respectively. These results are reasonable because the noisy cases in train1 increase the error rate.

Table 1. Summary of Data Sets

Data set	Usage	Number of blocks	Number of faulty blocks	Characteristics
train1	training data set	300	32	Ambiguous blocks exist in data set.
test1	test data set	300	34	
train2	training data set	200	23	Ambiguous blocks are removed in data set.
test2	test data set	200	23	

3.2 Attribute Selection

Because of the negative effect of irrelevant attributes in input data on classification algorithms, it is common to precede training with an attribute selection process that eliminates the unsuitable attributes. Reducing the dimensionality of the data by deleting irrelevant or redundant attributes improves the performance of training algorithms. Though the dimension of our input data is too low to be reduced, we perform attribute selection processes to get as many experimental results as possible. All methods supported by WEKA are used to get the best subset of the input attributes, and some of them are excluded because they select too many or too few attributes than is necessary with the two training data sets. The selected methods are CfsSubsetEval, ConsistencySubsetEval and PCA(Principal Component Analysis). CfsSubsetEval assesses the predictive ability of each attribute and the degree of redundancy among them. ConsistencySubsetEval evaluates attribute subsets by the level of consistency in the output values when the training instances are projected onto the subset [13]. The results of the attribute selection are shown in Table 2.

Table 2. Results of Attribute Selection

Training data set / Method	*train1*	*train2*
CfsSubsetEval	EBS, EBC, BS, BR	BRS, EBC, BS
ConsistencySubsetEval	EBS, BP, BS	BRS, EBS
PCA	BRS, EBS, EBC, BP	BRS, EBS, EBC, BP

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 7, July 2021

DOI 10.17148/IJARCCE.2021.10783

3.3 Training Results of RF Model

There are several measures to demonstrate the performance of prediction models, like Precision, Recall, F-measure and AUC. Prediction error information provides another way to analyze the prediction performance considering costs of mis-prediction. In the context of a binary classification model, two types of prediction error can occur, Type I and Type II. These are the two measures for evaluation of the prediction models in our study:

- Type I error rate:

number of non fault-prone blocks predicted as fault-prone / number of non fault-prone block

- Type II error rate:

number of fault-prone blocks predicted as non fault-prone / number of fault-prone blocks Type II error can be more serious than Type I error because it requires heavy costs at the later phases of system development. A Type II error may be detected during operations, leading to expensive repair work, whereas a Type I error is inspected prior to operations. Therefore, it is more important to minimize type II error.

Table 3. Results of RF Model Training

Model \ Training data set		train1		train2	
		Type I	Type II	Type I	Type II
RF (NotNormalize)	NotReduction	1/268	0/32	0/266	0/34
	CfsSubsetEval	0/268	1/32	0/266	0/34
	ConsistencySubsetEval	0/268	1/32	0/266	0/34
	PCA	2/268	0/32	0/266	0/34
RF (Normalize)	NotReduction	1/268	0/32	0/266	0/34
	CfsSubsetEval	0/268	1/32	0/266	0/34
	ConsistencySubsetEval	0/268	1/32	0/266	0/34
	PCA	1/268	0/32	0/266	0/34

Table 3 shows the training error results of the RF model. A prediction model with NotReduction means a model using input/output data whose dimension is not reduced by attribute selection methods. A model with Normalize means that input/output data of the model are normalized to the interval [0, 1], while NotNormalize means that raw data are used. As shown in the table, RF-Normalize have almost the same training results with RF- NotNormalize. This means that data normalization does not affect how well RF models are trained. Also, similar results are observed in the comparison of the RF models with and without attribute selection. Thus, in our experiment, attribute selection process does not help in increasing training performance. This result is partly caused by the low dimension of input data. RF shows very good training performance especially with train2, in that case it is trained with 0 errors in all data sets. The best result when train1 is used appears in the RF- NotReduction, with a Type I error rate of 1/268 and a Type II error rate is 0/32.

3.4 Testing Results of RF Model

The RF models trained with train1 and train2 are tested with test1 and test2. There are four types of testing results according to the used training data sets and the test data sets, and they are summarized in Table 4.

train1-test1: RF-NotReduction has the smallest Type II errors, 4, and NotNormalize case shows slightly better results than Normalize case in Type I error.

train1-test2: This case is related to generality of a prediction model because a general model trained with noisy data is tested with non-noisy test data. Considering Type II error, RF- NotReduction and RF-PCA have 0 errors, whereas others have 1~2 errors. RF-PCA shows good results, but it has too many Type I errors. In RF-NotReduction model, NotNormalize case shows slightly better results than Normalize case in Type I error, but the difference is negligible.

train2-test1: The case using training data without noises and test data with noises is meaningless because a prediction model trained with non-noisy data cannot understand noisy data distribution. RF-NotReduction shows slightly better result than other attribute selection cases.

train2-test2: This case makes few prediction errors because the data used for training and testing do not have ambiguous(noisy) blocks. Therefore, almost every model has 0 errors.

As a result, RF-NotReduction shows better results than other RF models, and the attribute selection methods and normalization make no difference to the performance of the prediction model.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 7, July 2021

DOI 10.17148/IJARCCE.2021.10783

Table 4. Result of RF Model Testing

Model		Test data set	*test1* Type I	*test1* Type II	*test2* Type I	*test2* Type II
RF (NotNormalize)	NotReduction	train1	3/177	4/23	0/177	0/23
		train2	5/177	11/23	0/177	0/23
	CfsSubsetEval	train1	4/177	9/23	1/177	1/23
		train2	7/177	12/23	0/177	1/23
	ConsistencySubsetEval	train1	3/177	9/23	2/177	2/23
		train2	6/177	11/23	0/177	0/23
	PCA	train1	4/177	5/23	5/177	0/23
		train2	9/177	11/23	0/177	0/23
RF (Normalize)	NotReduction	train1	1/177	4/23	2/177	0/23
		train2	5/177	10/23	0/177	0/23
	CfsSubsetEval	train1	1/177	10/23	1/177	2/23
		train2	7/177	12/23	0/177	1/23
	ConsistencySubsetEval	train1	3/177	9/23	2/177	2/23
		train2	6/177	11/23	0/177	0/23
	PCA	train1	4/177	5/23	4/177	0/23
		train2	7/177	10/23	0/177	0/23

3.5 Performance Comparison with Other Models

From early studies to present, MLP with backpropagation algorithm and SVM with kernel functions have shown good performances in prediction of software fault-proneness. To evaluate the prediction performance of the RF model, we compare the results of the best RF model, not using attribute selection methods, with two SVM models using two kernel functions and a MLP model [9]. All compared models use normalized data and the SVM models use two kernel functions: Radial Basis and Polynomial.

Table 5. Performance Evaluation Against other Models

Model		Test data set	*test1* Type I	*test1* Type II	*test2* Type I	*test2* Type II
RF	NotReduction	*train1*	1/177	4/23	2/177	0/23
		train2	5/177	10/23	0/177	0/23
SVM	Radial Basis	*train1*	0/177	15/23	0/177	2/23
		train2	1/177	11/23	0/177	0/23
	Polynomial	*train1*	0/177	13/23	0/177	2/23
		train2	1/177	12/23	0/177	0/23
MLP		*train1*	2/177	4/23	4/177	0/23
		train2	6/177	11/23	0/177	0/23

The evaluation results of the RF model against other models are provided in Table 5. Two SVM models show similar prediction performances in the four train-test cases. As shown in the table, the RF model significantly outperforms the SVM models especially in the train1- test1 case, and is better or equal in other cases except the meaningless train2-test1 case. The RF model has similar performance with the MLP model in Type II error rate, but makes the better results in Type I error rates.

4. CONCLUSION

Software fault-proneness prediction model is very important in the development of large information systems because early identification of the trouble spots greatly reduces the cost of system development. In this study, we construct a prediction model using RF classifier. To demonstrate the prediction capability of the RF model, the prediction error results of the proposed model are compared to those of the well-known prediction models, MLP and SVM model. As a result, the RF model significantly outperforms the SVM model and slightly outperforms the MLP model.

ACKNOWLEDGEMENTS

This work was supported by the Sungshin Women's University Research Grant of 2012

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 10, Issue 7, July 2021
DOI 10.17148/IJARCCE.2021.10783

REFERENCES

[1] N. Fenton and N. Ohlsson, "Quantitative analysis of faults and failures in a complex software system", IEEE Trans. on Software Eng., vol. 26, no. 8, (2000), pp. 797–814.
[2] I. Gondra, "Applying machine learning to software fault-proneness prediction", Journal of Systems ansd Software, vol. 81, no. 2, (2008), pp. 186-195.
[3] S. Bibi, G. Tsoumakas, I. Stamelos and I. Vlahavas, "Regression via Classification applied on software defect estimation", Expert Systems with Applications, vol. 34, no. 3, (2008), pp. 2091-2101.
[4] S. Lessmann, B. Baesens, C. Mues and S. Pietsch. "Benchmarking classification models for software defect prediction: A proposed framework and novel findings", IEEE Trans. on Software Eng., vol. 34, no. 4, (2008), pp. 485–496.

