Advances in Computer Science and Information Technology (ACSIT)
p-ISSN: 2393-9907; e-ISSN: 2393-9915; Volume 2, Issue 13; July-September, 2015 pp. 99-103

http://www.krishisanskriti.org/acsit.html

Analyzing Machine Learning and Statistical Models for Software Change Prediction

Ankita Bansal[1], Ayushee Sachan[2] and Manpreet Kaur[3]

[1,2,3]Netaji Subhash Institute Of Technology, Delhi, India
E-mail: [1]ankita.bansal06@gmail.com, [2]ayusheesachan14@gmail.com, [3]mkaur9326@gmail.com

Abstract—*Changes in software are unavoidable due to multiple reasons such as change in user requirements, change in technology, increasing customer demands etc. Introduction of such changes may adversely affect the quality of the software. Thus, a streamlined approach is required to identify the classes which are more change prone than others during early stages of software development life cycle. Identification of change prone classes allows managers to pay focused attention on such classes and thus, minimizes the effort required at the maintenance phase. This will inturn lead to efficient utilization of resources in terms of time, money and manpower. In this study, we have constructed various machine learning and statistical models for predicting change prone classes in the early phases of software development life cycle. The machine learning techniques used are primarily ensemble learners which will allow us to explore their use in the field of change prediction. For the purpose of empirical validation, three open source software systems, Apache projects (Abdera, Poi and Rave) are used. The results show a comparative performance of machine learning techniques and statistical models. Among the machine learning models, random forest and logitboost models have outperformed all the other models for all the Apache projects. Thus, this consistent result shows that the researchers and practitioners may use these models for prediction of change prone classes of similar Apache software.*

1. INTRODUCTION

Change is inevitable at all the stages of a software project [1]. With the increasing complexity of the software evolution, prediction of software changes is also becoming complex. Implementing changes at a later stage degrades the quality of a system and reduces its maintainability but is necessary for the real world environment [2-3]. Thus, among number of quality attributes, we are focusing on the attribute 'change proneness' in this paper. Prediction of the change prone classes if done in an initial stage allows the managers, developers, testers and designers to pay focused attention on such classes leading to saving of lots of resources [4-5]. This will inturn improve the quality of a software project. Thus, software change prediction is an essential activity for the software quality to be high [6].

Object oriented methodologies are increasingly being adopted to evaluate the quality of software [7]. Thus, we have used various object oriented metrics for constructing machine learning and statistical models which can be used for predicting change prone classes. The machine learning techniques used are primarily ensemble learners (EL); Bagging, LogitBoost, Adaboot and Random Forest. Despite the various advantages of ensemble learners such as their generalization ability, higher prediction accuracy etc., they are not being explored in the field of change prediction. Thus, in this paper, we aim to construct prediction models using these ensemble learners. Besides this, we have also compared the performance of these ensemble learner models with the traditional statistical model, logistic regression.

In this paper, we aim to address the following two research questions:

RQ1: Among multiple EL discussed, which ensemble learner is superior over all others and can be used for predicting change prone classes?

RQ2: How does the performance of EL compare with the statistical technique?

For the purpose of empirical validation, we have used 3 open source Apache projects, (Abdera, Rave and Poi). The performance of the models is evaluated using different performance measures and validation technique used 10-cross validation. We obtained consistent results across all the projects. For all the projects, Random Forest and Logitboost have shown superior performance than other classifiers. Thus, we recommend the use of these classifiers for prediction of change prone classes of similar Apache projects.

This paper is structured as follows: Section 2 summarizes the research background. Section 3 explains the research methodology. Section 4 shows the results. The last section concludes the paper.

2. RESOURCE BACKGROUND

In this section, we will focus on the datasets used, their change collection process and the variables used.

2.1 Empirical Data Collection

In this section, we have explained the datasets used in our study. Three open source Apache projects are used, namely

Abdera, Poi and Rave. All these projects are written in Java programming language. Apache Abdera is a Java implementation of the atom syndication and atom publishing protocol (https://**abdera**.**apache**.org/). Apache Poi provides pure Java libraries for Microsoft office formals (https://**poi**.**apache**.org/) and Apache Rave aggregates and serves web widgets(https://**rave**.**apache**.org/). Open source software is publically accessible, changed and shared. Source code of these software systems is available online. We have compared two successive versions of each of these software to identify those classes in the previous version which have changed in the next version. The versions used for each of the software are listed in table 1.

We briefly explain the change collection process in this section.

Change Collection Process

The classes with the same name in the two successive versions are extracted and compared. The change is calculated as the number of lines of code added, deleted and modified in the class of present version with respect to the same class in the previous version.

The 'TOTAL CHANGE' for a class as follows:

- Each added or deleted line is counted as one SLOC change.
- Each modified line is counted as two SLOC change i.e. one deletion followed by one addition.

We defined a binary variable 'CHANGE', which is assigned a value 1 if 'TOTAL CHANGE' >1 or 0 otherwise.
The number of common classes between the two successive versions and the classes changed in provided in table 1.

Table 1: Software Details.

Software	Total Number of Classes	Classes Changed	Versions	
			Previous Version	Current Version
Abdera	685	635	1.1.2	1.1.3
Poi	2786	2706	3.9	3.10
Rave	685	225	0.22	0.23

2.2 Variables Used

In this section, we explain the independent and dependent variables used in this study.

The independent variables used are various object oriented metrics. We have used the popular Chidamber and Kemerer [2] metrics along with the metric used to measure the size of the software (LOC).

Table 2: Definition of Independent Variables.

METRIC		DESCRIPTION
DIT	Depth of Inheritance Tree	The maximum number of edges between a given class and a root class.
RFC	Response for a Class	Aggregate of local methods of a class and all the external methods directly called by any local method.
CBO	Coupling between Objects	The number of external classes whose attributes and methods are used from the measured class.
LOC	Lines of Code	The lines of code the class contains.
WMC	Weighted Methods Per Class	Count of sum of complexities of all methods in a class.
LCOM	Lack of Cohesion	For each data field in a class, the percentage of the methods in the class using that data field. The average of percentages is subtracted from 100%.

Change proneness is the dependent variable in our study. If a class in changed in the next version of the software, it is called as change prone and not-change prone otherwise. Class changes are induced either by class itself or due to changes in other classes. Internal changes can be identified from the source code and are calculated interms of number of lines added, deleted and modified in the present version with respect to the previous version. We are dealing with internal changes in this study.

3. RESOURCE METHODOLOGY

In this section, we explain the various data analysis methods and the performance measures used to evaluate the models.

3.1 Data Analysis Method Used

In this section, we explain the various data analysis methods used and the performance measures used to evaluate the models.Models for predicting change proneness of object oriented software can be categorized under two categories, namely – statistical and machine learning models.

3.1.1 Statistical Model: Logistic Regression (LR)

Logistic Regression is applied on independent variable set to predict dependent variable ([8-9]). It is applied when dependent variable is likely to have two observed outcome.

3.1.2 Machine Learning Model

Brief description of various Machine Learning techniques is given in this section. WEKA tool, open source data analysis software developed by the University of Waikato [10] has been used with its default setting.

Random Forest

Random Forest is a method for classification and regression, proposed by Brieman [11]. It is a combination of trees predictors such that each tree depends on the value of a random vector. Thus, a Random Forest is a classifier containing many decision trees. The forest chooses the classification having the most votes, i.e. the majority wins.

Linear models can also be used as the base estimator in random forest instead of decision trees [12].

Bagging

Bagging (also known as Bootstrap aggregation) is one of the machine learning ensemble meta-algorithms proposed by Breiman in 1994 [13]. It is designed to improve the stability and accuracy of the classification models by creating various versions of the training sets. Various homogenous/comparable training sets are framed and a newly formed function is trained for each of them. Result of class prediction is based on model average approach [14]. Bagging helps in reducing variance and avoids over fitting.

Boosting

Boosting is a machine learning algorithm which is used for reducing variance in supervised learning. It is a method for improving correctness of a given learning algorithm by incorporating simple rules to form an ensemble such that performance of single member is boosted. Logitboost, Adaboost are some of the various boosting algorithm available. The basic difference between various boosting algorithms is their hypothesis and significance of training data points. Adaboost introduced in 1995 by Freud and Schapire [15] is the most accepted algorithm among all boosting algorithms. Weak learning algorithm is called in a series of rounds, n=1……,N. At the beginning all the weights are set equal and on each round, weights of incorrect classified example are increased whereas weights of correct classified examples are decreased such that focus is on hard examples [15].

Correlation based features selection (CFS)

Feature selection is a basic principle of machine learning techniques. In this method relevant features are preferred for model construction. Features are characterized as relevant if the correlation between independent variables and dependent variables exist but they should not have any resemblance among each other. Removal of redundant features does not lead to data loss [16]. We have applied CFS which is provided in WEKA tool [10] for data reduction. Independent variables which are necessary for the data are selected [16-17]. Advantages of feature selection are less execution time, more understandability, accurate, better prediction accuracy [17].

3.2 Performance Measures Used

Let a = number of classes correctly predicted to be change prone (true positive).

b = number of classes incorrectly predicted to be non- change prone (false negative).

c = number of classes incorrectly predicted to be change prone (false positive).

d = number of classes correctly predicted to be non- change prone (true negative).

Different performance measures can be defined as follows:

Sensitivity

Sensitivity is the ratio of correctly predicted change prone classes by total change prone classes.

Mathematically, sensitivity = a / (a + b)

Specificity

Specificity is the ratio of correctly predicted non change prone classes by total non-change prone classes.

Mathematically, specificity = d / (c + d)

Accuracy/Precision

Accuracy is defined as the ratio of number of classes (including both change and not change prone) that are predicted correctly by the total number of classes.

Mathematically, accuracy= a+d / (a+ b + c+ d)

Receiver operating characteristics curve

In a receiver operating characteristic curve (ROC), the true positive rate i.e. sensitivity is plotted on y-axis the false rate i.e. (1-specificity) is plotted on x-axis. Different cut-off points between 0 and 1 are taken, then sensitivity and specificity values at each cut off point are calculated and ROC curve is constructed with the help of these values. Among a number of cut-off points, the cut-off point where sensitivity equals specificity is known as optimal cut-off point. Thus, the sensitivity and specificity values obtained at the optimal cut-off point are considered for model evaluation. Besides this, the area under the ROC curve (AUC) is also used to measure the performance of the models. Higher the values of sensitivity, specificity and AUC, better the performance of the model.

Cross validation

The results are highly optimistic if the model is tested on the same dataset as on which it is trained. Thus, it is very important to use different training and testing sets. For this purpose, we have used K-cross validation (the value of K is 10), where the dataset is divided into 10 equal parts. 9 parts are used for training the data and the remaining 1 part is used for testing the data. This process is repeated until each of the 10 parts is used as testing sets.

4. RESULT ANALYSIS

Various models using different ensemble learning techniques are constructed for prediction of change prone classes. Before model construction, CFS technique is applied to obtain a subset of metrics (independent variables) which are significant predictors of change proneness. Table 3 lists the significant metrics of each of the three datasets, obtained after applying CFS.

Table 3: Datasets Used.

Datasets Used	Significant Metrics
Apache Abdera	WMC, DIT, LOC
Apache Poi	WMC, RFC, LOC
Apache Rave	WMC, RFC, LOC

We now discuss the results of validation presented in tables 4, 5 and 6 of Abdera, Poi and Rave respectively.

Table 4 shows that for Abdera dataset, the highest AUC, sensitivity and specificity are given by Random Forest model. The second best performance is given by LogitBoost with AUC 0.7, sensitivity and specificity of 0.65 and 0.68 respectively.

Table 4: Validation Results of Apache Abdera.

Method Used	Sensitivity	Specificity	AUC
Logistic Regression	0.67	0.70	0.62
LogitBoost	0.65	0.70	0.70
AdaBoost	0.61	0.60	0.65
Bagging	0.70	0.65	0.66
Random Forest	0.71	0.72	0.71

In Apache Poi (table 5), LogitBoost outperformed all other models with the highest values of AUC, specificity and sensitivity. Random Forest follows LogitBoost with AUC of 0.63 which is comparable to Bagging. But latter has quite low sensitivity and specificity.

Table 5: Validation Results of Apache Poi.

Method Used	Sensitivity	Specificity	AUC
Logistic Regression	0.50	0.60	0.52
LogitBoost	0.61	0.62	0.66
AdaBoost	0.70	0.60	0.60
Bagging	0.60	0.60	0.63
Random Forest	0.61	0.63	0.63

In Apache Rave (table 6) Random Forest has the maximum AUC i.e. 0.69 with high sensitivity and specificity of 0.66 and 0.62 respectively. The AUC results of Bagging and LogitBoost are comparable with values of 0.65 and 0.64 respectively.

We have used one traditional method i.e. logistic regression besides the ensemble learning techniques. We observe that the AUC of the logistic regression is comparable to the ensemble learning techniques. The former has comparable values of sensitivity and specificity besides AUC with the latter.

Table 6: Validation Results of Apache Rave.

Method Used	Sensitivity	Specificity	AUC
Logistic Regression	0.64	0.61	0.64
LogitBoost	0.61	0.61	0.64
AdaBoost	0.61	0.60	0.61
Bagging	0.64	0.62	0.65
Random Forest	0.66	0.62	0.69

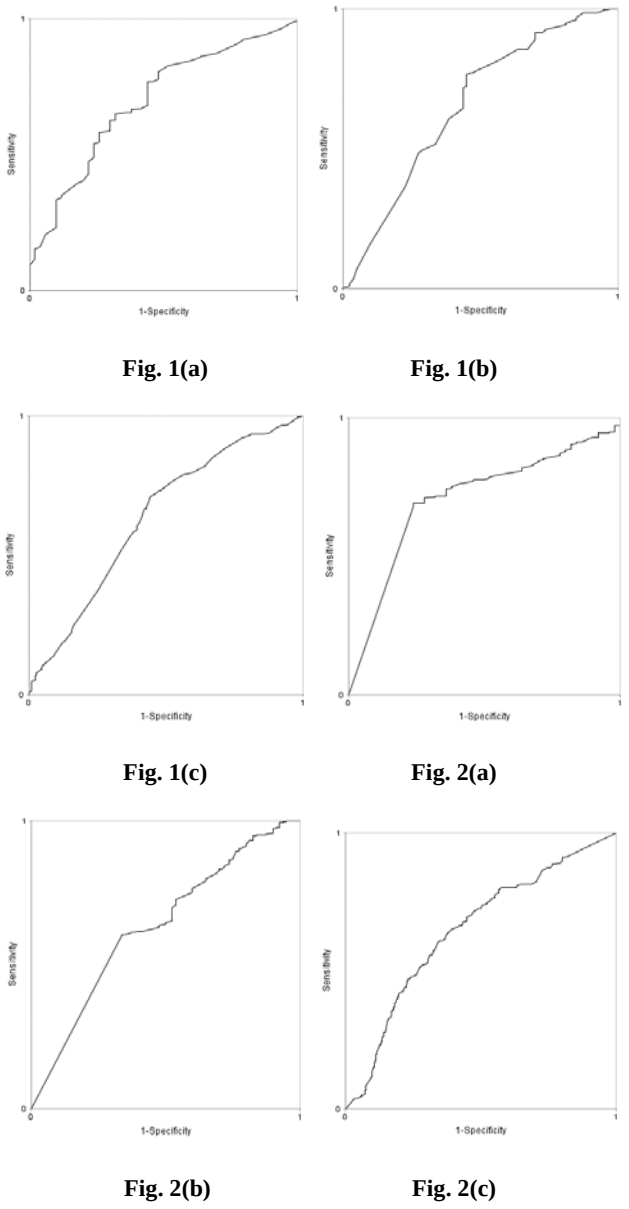

Fig. 1(a) **Fig. 1(b)**

Fig. 1(c) **Fig. 2(a)**

Fig. 2(b) **Fig. 2(c)**

Fig. 1: ROC curve of LogitBoost for (a)Abdera, (b)Poi, (c)Rave
Fig. 2: ROC curve of Random Forest for (a)Abdera, (b)Poi, (c)Rave

Overall, we conclude that among ensemble learners, random forest and LogitBoost have outperformed all other classifiers. Their ROC curves obtained using all the datasets are shown in Fig. 1 and 2. Also, the statistical model has shown comparable and competitive performance with the ensemble learners. Thus, the researchers and practitioners could use random forest and LogitBoost models for prediction of change prone classes for similar open source Apache Software Systems.

5. CONCLUSION AND FUTURE WORK

This study focuses on identifying change prone classes during early stages of software development life cycle which will lead to substantial saving of resources. The relationship between object oriented metrics and change proneness using ensemble learners and statistical method is constructed using various models. The datasets (LogitBoost, AdaBoost, Bagging and Random Forest) used are three open source Apache software systems (Abdera, Poi and Rave). Two versions of each of these are analyzed to predict the change prone classes.

CFS technique is used to obtain a significant subset of independent variables for predicting the change prone classes. For all the three datasets, we found that WMC and LOC are significant in predictors of change proneness. Thus, researchers and practitioners may use these metrics for model construction. Among various ensemble learning techniques Random Forest and LogitBoost gave the best results of AUC which are (0.71, 0.70), (0.63, 0.66) and (0.69, 0.64) for Apache Abdera, Poi and Rave respectively. Thus, we recommend the use of these classifiers for prediction of change prone classes of similar Apache projects.

Besides this, we also concluded that the performance of the statisitcal method (logistic regression) is comparable to the performance of ensemble learners.

We plan to replicate this study on more number of large datasets which will allow us to make our results more generalizable. We also plan to carry out cross project predicting where different projects are taken for testing and training.

REFERENCES

[1] Donaldson, S. E., and Siegel S. G., "Successful Software Development", *Prentice Hall Professional*, 2, pp. 188.

[2] Chidamber, S.R., and Kemerer, C.F., "A metrics suite for object oriented design", *IEEE Transactions on Software Engineering*, 20, 6, 1994, pp. 476–49.

[3] Li, W., and Henry, S., "Object-oriented metrics that predict maintainability" , *Journal of Systems and Software - Special issue on object-oriented software*, 1993, pp. 111–122.

[4] Harter, D. E., and Slaughter, S. A., "Quality Improvement and Infrastructure Activity Costs in Software Development: A Longitudinal Analysis", *Journal of Management Science*, 2003, pp. 784-800.

[5] Aggarwal, K.K., Singh, Y., and Kaur, A., Malhotra, R., "Empirical Study of Object-Oriented Metrics", *Journal of Object Technology*, 5, 2006 , pp. 149-173.

[6] Chaumum, M.A., and K. H., "A change impact model for changeability assessment in object oriented software systems" *Third European conference on software maintenance and reengineering*, 1999.

[7] Malhotra, R., and Khanna, M., "Investigation of relationship between object oriented metrics and change proneness", *International journal of machine learning and Cybernetics, Springer- Verlag*, 2012, pp. 273-286.

[8] Basili, V., Briand, L., and Melo, W., "A validation of object-oriented design metrics as quality Indicators", *IEEE Transactions on Software Engineering*, 1996, pp. 751–761.

[9] Hosmer, D., and Lemeshow, S., "Applied logistic regression", New York, 1989.

[10] Weka. Available: http://www.cs.waikato.ac.nz/ml/weka/.

[11] L. Breiman, "Random forests," *Machine Learning*, 45, October 2011, pp. 5–32.

[12] Breiman, L., "Bagging Predictors", *Technical Report No. 421, Partially supported by NSF grant DMS-9212419*, 1994.

[13] Bagging, Available on: https://en.wikipedia.org/wiki/Bootstrap_aggregating.

[14] Freund, Y., and Schapire, R.E., "A Short Introduction to Boosting", *Journal of Japanese Society for Artificial Intelligence*, 1999, pp. 771-780.

[15] Hall, M.A., "Correlation-based feature selection for discrete and numeric class machine learning", *In Proceedings of the 17th International Conference on Machine Learning*, 1999, pp. 359-366.

[16] Malhotra, R., Kaur, A., and Singh, Y., "Application of Machine Learning Methods for Software Effort Prediction", *In Newsletter ACMSIGSOFT Software Enginnering Notes*, 2010.

[17] Michalak, K., and Kwasnicka, H., "Correlation-based feature selection strategy in neural classification", *Sixth international conference on intelligent systems design and applications*, 2006, pp. 741–746.