# Chapter 86
# Prediction of Change-Prone Classes Using Machine Learning and Statistical Techniques

**LinRuchika Malhotra**
*Delhi Technological University, India*

**Ankita Jain Bansal**
*Delhi Technological University, India*

## ABSTRACT

*For software development, availability of resources is limited, thereby necessitating efficient and effective utilization of resources. This can be achieved through prediction of key attributes, which affect software quality such as fault proneness, change proneness, effort, maintainability, etc. The primary aim of this chapter is to investigate the relationship between object-oriented metrics and change proneness. Predicting the classes that are prone to changes can help in maintenance and testing. Developers can focus on the classes that are more change prone by appropriately allocating resources. This will help in reducing costs associated with software maintenance activities. The authors have constructed models to predict change proneness using various machine-learning methods and one statistical method. They have evaluated and compared the performance of these methods. The proposed models are validated using open source software, Frinika, and the results are evaluated using Receiver Operating Characteristic (ROC) analysis. The study shows that machine-learning methods are more efficient than regression techniques. Among the machine-learning methods, boosting technique (i.e. Logitboost) outperformed all the other models. Thus, the authors conclude that the developed models can be used to predict the change proneness of classes, leading to improved software quality.*

## INTRODUCTION

A number of studies have empirically validated the relationship between object oriented metrics and important external attributes such as reliability, effort, fault proneness, change proneness, etc. (Aggarwal, Singh, Kaur & Malhotra, 2009; Singh, Kaur & Malhotra, 2010; Gyimothy, Ferenc & Siket, 2005; Bieman, Andrews & Yang, 2003; Tsantalis, Chatzigeorgiou & Stephanides, 2005; Li & Henry, 1993;

DOI: 10.4018/978-1-5225-3923-0.ch086

Briand, Wust & Lounis, 2001). This has been done to determine whether object oriented metrics are useful quality indicators. In this chapter, we have investigated the relationship between object oriented metrics and change proneness. Every software undergoes number of changes throughout its life period - to improve functionality, to fix bugs, to add new features etc. Additionally, requirements of the user may change with time, leading to further changes in the software. This may result in various versions of a software. But making changes in a particular version of the software is not an easy task and requires large amount of resources in terms of money, time, and manpower. This is because software typically consists of a large number of classes. It might be possible that a single change in a class is propagated to other classes, which in turn will lead to change in the classes affected by the change. As a result, significant percentage of the classes may need to be changed. It has been studied that the largest percentage of the software development effort is spent on rework and maintenance. Thus, it would be highly beneficial if we get to know the classes which are prone to changes. This will help developers as they can concentrate on these change prone classes and make a more flexible software by modifying the classes which are more prone to changes. Developers can take focused preventive actions which will help to reduce the maintenance costs and improve quality. Also, developers can allocate resources more judiciously. Besides these advantages, we also get insight about the design of the software by correctly predicting the change prone classes, e.g., if a change in a particular class has a large impact on some other class, then we can conclude there is high coupling between the two classes and thus to improve the design, coupling must be reduced.

The aim of this chapter is to establish a relationship between object oriented metrics (Li, Henry, Kafura & Schulman, 1995; Chidamber & Kemerer, 1994; Lorenz & Kidd, 1994) and change proneness using various machine learning techniques i.e. adaboost, logitboost, naivebayes, bayesnet and J48, and one traditional statistical method i.e. logistic regression. We have also compared the performance of the machine learning techniques and statistical method. The empirical validation is carried out on an open source software, Frinika, written in java language. Two versions of the software are taken and analyzed for changes. The results are evaluated using Receiver operating characteristic curve (ROC) curve by measuring area under the curve (AUC).

The rest of the chapter is organized as follows: Section 2 reviews the related work focusing on the key points in the domain. Section 3 explains the independent and dependent variables used in our study and various evaluation measures used. Section 4 discusses the research methodology used to develop the model. Section 5 summarizes the results and finally the work is concluded in section 6.

## LITERATURE REVIEW

A software undergoes number of changes to enhance its functionality. Various researchers have devised techniques to identify change prone classes. The paper (Bieman, Andrews & Yang, 2003) has identified change prone classes and then distinguished between the local change proneness and change proneness that occur due to interactions with other classes. The paper (Tsantalis, Chatzigeorgiou & Stephanides, 2005) has devised a technique to find the probability that each class will change in future generation. In other words, they have found the probability of change in a class A which is affected by another change prone class B. Similar type of work was done by A.R. Sharafat et al. (Sharafat & Tavildari, 2007). They have calculated three probabilities for each class: $P_{ic}$ (internal change probability), $a_{ji}$ (propagation probability) and $P_{is}$ (total probability of change). Besides these probabilities, one more history based time