J Inf Process Syst, Vol.13, No.4, pp.778~804, August 2017
https://doi.org/10.3745/JIPS.04.0013

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

# Prediction & Assessment of Change Prone Classes Using Statistical & Machine Learning Techniques

Ruchika Malhotra* and Ravi Jangra*

**Abstract**

Software today has become an inseparable part of our life. In order to achieve the ever demanding needs of customers, it has to rapidly evolve and include a number of changes. In this paper, our aim is to study the relationship of object oriented metrics with change proneness attribute of a class. Prediction models based on this study can help us in identifying change prone classes of a software. We can then focus our efforts on these change prone classes during testing to yield a better quality software. Previously, researchers have used statistical methods for predicting change prone classes. But machine learning methods are rarely used for identification of change prone classes. In our study, we evaluate and compare the performances of ten machine learning methods with the statistical method. This evaluation is based on two open source software systems developed in Java language. We also validated the developed prediction models using other software data set in the same domain (3D modelling). The performance of the predicted models was evaluated using receiver operating characteristic analysis. The results indicate that the machine learning methods are at par with the statistical method for prediction of change prone classes. Another analysis showed that the models constructed for a software can also be used to predict change prone nature of classes of another software in the same domain. This study would help developers in performing effective regression testing at low cost and effort. It will also help the developers to design an effective model that results in less change prone classes, hence better maintenance.

**Keywords**

Change Proneness, Empirical Validation, Machine Learning Techniques, Software Quality

## 1. Introduction

The software industry has grown considerably faster than any other industry. Software is evolving with time to incorporate newer demands or to amend any present defects. The cycle of change is becoming shorter as technology progresses. The effort required from development to testing has increased manifold with this time crunch. Resources, such as time, cost, and effort, are limited in the software industry while working to develop and maintain a software. A lot of research has been done to study the relationship between object-oriented (OO) metrics and various quality attributes of a software like fault proneness and maintainability [1-7]. The prediction of these software attributes helps in identifying the weak classes of software. Thus, we can effectively plan a project's constraint resources by assigning ap-



Manuscript received November 21, 2013; first revision May 29, 2014; second revision September 11, 2014; accepted October 28, 2014; onlinefirst January 28, 2015.

**Corresponding Author:** Ruchika Malhotra (ruchikamalhotra2004@yahoo.com)

* Dept. of Software Engineering, Delhi Technological University, New Delhi, India (ruchikamalhotra2004@yahoo.com, ravi_jangra25@yahoo.co.in)

propriate resources to the weak classes of software, so that these classes are better maintained and tested.

Maintenance accounts for about 40%–70% of the total cost of the software. Thus, it is critical to evaluate the probability of change in a particular part of the software (change proneness). The prediction of change prone classes can help in optimizing the maintenance and testing phases of a software project. Change prone classes must be thoroughly tested and tracked as the software evolves. This would lead to better quality software with less defects and changes. It would also lead to the effective and economic utilization of resources by focusing resources on these classes. The identification of change prone classes in the early phases of the software development life cycle also aids in the drastic reduction of maintenance costs. This is because the sooner an error is detected in a product, the fewer amount of resources there are that are consumed to correct that error. Otherwise, the cost to correct an error increases exponentially in each undetected phase of the software's life cycle. Thus, the development of models that predict change prone classes aids software developers in delivering a better quality product at optimum costs, as minimum changes and errors are propagated to the later stages of software development.

In order to develop change prediction models, researchers need to assess the effectiveness of a number of methods. These methods could be traditional statistical methods or machine learning methods. It is important to explore a number of methods as different methods may give varied results on different data sets. Thus, the search for a method that develops an effective model on a number of data sets continues. Apart from using different methods, it is crucial to evaluate whether a model that has been developed using the training data from a specific data set can provide good prediction results for another data set. This type of procedure is known as inter project validation. In order to develop inter-project models, we need to train the model on a specific data set and provide a different data set for validating the model. The development of an inter project model is useful, due to the fact that it saves a lot of effort. The training data for each specific data set may not be available and may be scarce in nature. As such, reusing the training data for another software data set saves a lot of time, as well as effort.

The aim of this study is to develop effective change prediction models that would help in ascertaining the change prone classes in a software. The various objectives of the study are as follows:

1) To perform a comprehensive study to investigate the relationship between OO metrics and the change proneness of classes.
2) To analyze and assess the capability of various machine learning and statistical methods for developing change prediction models.
3) To evaluate the effectiveness of inter-project models and to determine if they are practical and useful. The study determines whether a model trained on a specific software data set can be used for predicting the change prone classes of another data set.

In order to explore the relationship between various OO metrics and change prone classes, we developed change prediction models using the two open source software data sets of Art-of-Illusion (AOI) and Sweet-Home-3D, which were developed in the Java language. Two versions of each software system was taken and analyzed for changes. The changes were analyzed in binary form. If any line was added, deleted, or modified from among the two versions it was then marked as a change. The OO metrics were calculated for classes that are common to both versions of the software and were combined with change statistics that were calculated beforehand, to serve as data points for further analysis. Various machine learning methods and the logistic regression method (a statistical method) were analyzed for their performance in predicting the change prone classes of a software. The results show that the ma-

chine learning methods perform well in predicting change prone classes and that they are on par with the statistical methods. We also performed a statistical test (t-test) to test the hypothesis of whether various methods are statistically different from each other for developing change prediction models.

It is very intuitive for researchers to validate the results from the same data set from which it was derived from. Hence, for validation purposes we took two versions of open source software (AOI and Sweet-Home-3D), which were written in the Java language. However, we also used the Sweet-Home-3D software system data set as a test set for the inter-project validation of the change proneness prediction model, which was developed using training data from AOI software. The performance of various models was assessed using receiver operating characteristic (ROC) analysis. The results of this study are phenomenal and are helpful for allowing testers and developers to effectively utilize their resources during the maintenance and testing phases.

This paper is organized as follows: Section 2 summarizes the related work in this field. Section 3 summarizes the research methodology used in the study; along with the different variables of the study, research hypothesis, and data collection procedure. Section 4 describes the experimental framework with a description of all the methods used in the study, along with the performance measures and model building techniques. Section 5 gives the results of the study and Section 6 presents the threats to validity in this work. Section 7 provides how the work presented in this study can be applied and the conclusion is given in Section 8.

## 2. Related Work

Software keeps on evolving with time, which leads to changes in the functionality of the system or to the removal of defects that were previously present in it. Prediction models for identifying change prone classes in the software can help to resolve these issues with ease. Various models have been proposed in this field using machine learning techniques and statistical techniques. Han et al. [8] used the behavioral dependency measurement (BDM), which is a method used for rating classes according to the probability of change, in order to predict change proneness in UML 2.0 models. BDM was found to be an effective indicator of change proneness.

Ajrnal Chaumum et al. [9] carried out a study on the telecommunication system to analyze the impact of a change. They defined a change impact model and mapped it to C++ language to study the consequences of changes made to the classes of a system. Their result showed that the design of a system defines the system's reaction to incoming changes, and that well-chosen OO metrics can function as indicators of changeability.

When we add or modify a class it changes the OO design of the system. Tsantalis et al. [10] analyzed this effect on change proneness in their work. Their work categorizes three axes, namely change inheritance, reference, and dependency. Probabilities were extracted from two open source multi-version software projects (JFlex and JMol) for studying the stability of design over successive versions of the software project and to identify a level beyond which the remodeling of a design cannot be pursued. They performed statistical analysis on the projects. The results of their analysis revealed an improved correlation between extracted probabilities and actual changes in each class, in comparison to the prediction model that relies upon past data.

Sharafat and Tavildari [11] used a reverse engineering technique to collect code metrics for various

versions of a software project (JFlex). They proposed a probabilistic model that predicts the probability of a change in each class. This approach is based upon the change history and source code of the class.

Zhou et al. [12] found the effect of size on the associations between OO metrics and change proneness of a class. They considered the following three size metrics: SLOC, which is available only after the implementation stage has been completed; NMIMP (the number of methods implemented in a class); and NumPara (the sum of the parameters of these methods), which is available earlier during the high level design stage. They concluded that the confounding effect of size on the relationship between the metrics and change proneness exists and thus, should be removed to avoid misleading results. Another study by Lu et al. [1] has investigated the relationship between OO metrics and change proneness. The study considered a large number of OO metrics (62 metrics) and the empirical validation was conducted using a large number of systems (102 systems) so that they could generalize their results. However, both the studies by Zhou et al. [12] and Lu et al. [1] used statistical methods for developing change prediction models.

Our previous work [13] also established a relationship between OO metrics and change proneness. The study developed change prediction models using various machine learning (AdaBoost, LogitBoost, Naïve Bayes, Bayes Net, and J48) and statistical (logistic regression) methods. However, the dataset used is in the previous study is different than in this study. Frinika, which is a complete music workstation, was the type of dataset used in the previous study. Moreover, we only used a single dataset in our previous work. Whereas, in our present work, we have collected the changes of two completely different, open source software data sets (AOI and Sweet-Home-3D). In this work, we also performed inter-project validation. In other words, we have used one project for training the model and the other project for testing the model. This type of cross validation was also not done in our previous work.

The study is different from existent previous work as it analyzes and evaluates ten different machine learning and statistical methods for developing change prediction models. Even though certain researchers have developed change prediction models, these models only assess the capability of the widely

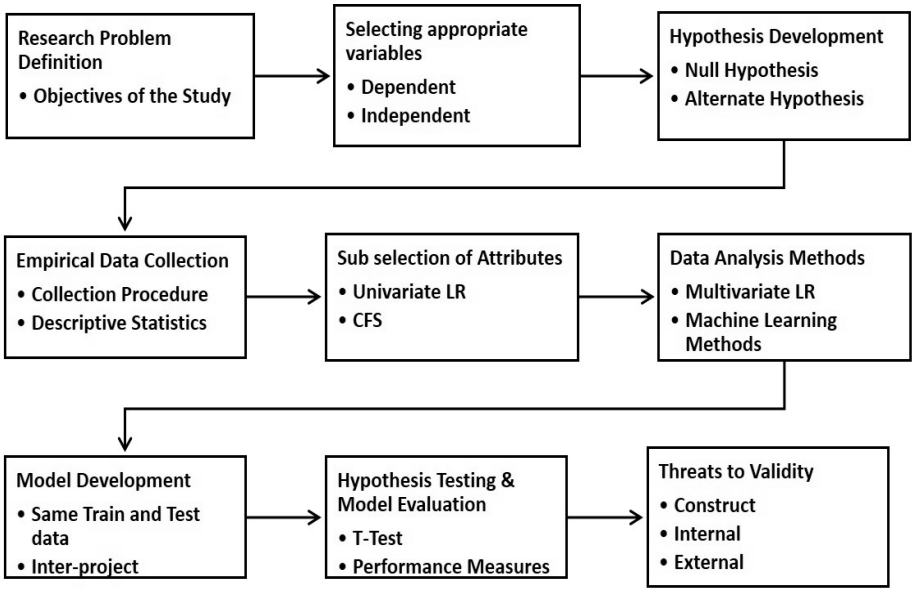


**Fig. 1.** Basic steps followed.

used statistical method (logistic regression) and very few machine learning methods. Moreover, we performed a statistical test (t-test) in our study to evaluate whether or not the pairwise comparison between the capabilities of different methods is statistically significant. This is important for providing definite conclusions about the capabilities of different methods. This study also explores the applicability of inter-project validation using machine learning methods. Inter-project validation saves a lot of effort and time as the training data of a project can be reused to predict the change prone classes of a different software project. Such models could be effectively used when the project has constraint resources and when the training data of a project is too expensive to collect and may not be easily available for developing effective models. Thus, this study has a significant impact on both academia and the software industry as it explores a variety of methods for change prediction and guides academia and the software industry in choosing an effective method. Moreover, the study also advocates the use of inter-project validation for change prediction, which is a novel technique for developing models in the field of change prediction.

# 3. Research Methodology

We will briefly explain the basic methodology and describe each step in detail in the subsequent sections. The methodology is explained via a diagram in Fig. 1. The various research steps involved in the study are as listed below.

- We first selected the dependent and the independent variables for this study. Section 3.1 states the variables of this study in detail. These variables were selected in order to explore the relationship between various OO metrics and change prone classes. In the next step, we state the hypothesis that we evaluated (Section 3.2). The hypothesis was developed in order to explore the second objective statistically.
- The data from both of the software data sets was collected from an open source repository named sourceforge.net. A detailed description of the data collection procedure and the details of the data sources are explained in Section 3.3. The descriptive statistics of both of the data sets used in this study are also provided in this section.
- All of the change prediction models were developed using the 13 metrics explained in Table 1. However, in order to develop an effective prediction model, we needed to reduce the dimensions of our input set. We did so by finding a subset of independent variables that were significant in predicting the dependent variable and change prone classes. We used univariate logistic regression (Section 4.1) for reducing the number of input variables, while developing a model using the statistical method and the correlation-based feature selection (CFS) technique for input variables reduction when using the machine learning methods. The CFS technique is explained in Section 4.2.1. The metrics that were selected after these methods were applied are presented in Section 5.1.
- We analyzed the change prediction models that were developed by using both the statistical and the machine learning methods. Brief descriptions of all of the methods that we used in this study are presented in Sections 4.1 and 4.2.
- Next, we constructed of various machine learning and statistical models to predict change proneness for both of the projects. The various models were developed with the aid of a WEKA tool. These models have been developed using ten-fold cross validation technique where the same soft-

ware is used for training and validation. We also evaluated inter-project models. A brief description of the model development techniques that we used is given in Section 4.3.

- All of the prediction models developed in this study were evaluated on three performance measures: sensitivity, specificity, and area under the ROC curve (AUC). All of which are briefly explained in Section 4.4. In order to validate the hypothesis, we used the t-test; the results of which are presented in Section 5.4.
- We performed inter-project validation using Sweet-Home-3D software as the test set and AOI software as the training set. These results are stated in Section 5.5.
- The various threats to the validity of this study were also evaluated. The three threats to validity of this study are as follows: construct validity, internal validity, and external validity. The threats are presented in Section 6.

## 3.1 Independent and Dependent Variables

The independent and dependent variables that were used in this study are described below.

### 3.1.1 Independent variables

Object-oriented metrics are used to quantify the various aspects of a software process. A single metric is not sufficient for studying all of the aspects related to a software. As such, a combination of a number of metrics needs to be used. The independent variables (i.e., various OO metrics) in this study are summarized in Table 1. The metrics that we included in our study account for various characteristics of a software like its coupling, size, cohesion, inheritance, etc. These metrics were evaluated for all of the classes of a software using the scientific tool of Understand [14,15].

### 3.1.2 Dependent variables

Rapid changes in the utility of a software require modifications and upgrades in the code of the software. Change proneness prediction would be a great help for low cost enhancements. This study aims at studying the relationship between OO metrics and the change prone attributes of classes in software. In this context, change is anything that is added, deleted, or modified in terms of SLOC. Thus, the dependent variable in our study is change proneness.

## 3.2 Research Hypothesis

We aimed to perform a statistical test to evaluate whether or not the differences amongst various methods is statistically significant with the help of t-test. In our hypothesis, we used the LogitBoost method as the baseline method. The hypothesis that we tested is as explained below.

- Null Hypothesis: The change prediction model developed using the LogitBoost technique is similar to the change prediction models that were developed using the other nine methods (Bagging, AdaBoost, LR, MLP, Bayes Net, Naïve Bayes, Random Forest, J48, and NNge).
- Alternate Hypothesis: The change prediction model developed using the LogitBoost technique is better than the change prediction models that were developed using other nine methods (Bagging, AdaBoost, LR, MLP, Bayes Net, Naïve Bayes, Random Forest, J48, and NNge).

**Table 1.** Independent variables

| S. No. | Metric | Definition | Source | Dimension |
|---|---|---|---|---|
| 1 | Number of base classes (NIB) | Number of immediate base classes | [16] | Inheritance |
| 2 | Coupling between objects (CBO) | Number of classes to which a class is coupled and vice versa | [17,18] | Coupling |
| 3 | Number of children (NOC) | Number of immediate subclasses of a class | [17,18] | Inheritance |
| 4 | Number of attribute (NOA) | Total number of attributes/variables defined in a class | [18,19] | Size |
| 5 | Response for a class (RFC) | Number of methods in the class including the methods accessible to an object class due to inheritance | [17,18] | Coupling |
| 6 | Number of private methods (NPRM) | Number of local (not inherited) private methods | [16] | Size |
| 7 | Number of protected methods (NPROM) | Number of local protected methods | [16] | Size |
| 8 | Number of public methods (NPM) | Number of local public methods | [20] | Size |
| 9 | Source lines of code (SLOC) | The number of lines that contain source code | [21] | Size |
| 10 | Depth of inheritance tree (DIT) | Maximum number of steps from the class node to the root node of the inheritance tree | [22,23] | Inheritance |
| 11 | Maximum nesting level (MNL) | Maximum nesting of control constructs | [16] | Inheritance |
| 12 | Lack of cohesion amongst methods (LCOM) | For each data field in a class, the percentage of the methods in the class using that data field; the percentages are averaged and then subtracted from 100% | [17,18,23] | Cohesion |
| 13 | Weighted methods per class (WMC) | Sum of complexities of all the methods in a class | [17,18] | Size |

## 3.3 Empirical Data Collection

This section presents a detailed description of our data sources and the data collection method that we used. We examined two open source software, which were written in Java, from the open source repository of sourceforge.net (http://sourceforge.net). The changes in each software were characterized by the changes in the classes of different versions of the corresponding software. Two stable versions of each software were analyzed by extracting the common classes that were present in both versions of the software. We then analyzed the changes in these common classes.

We investigated the AOI software, which is a full-featured 3D modeling, rendering, and animation studio that is available for all operating systems. The two stable versions evaluated for AOI in this study were v2.9.2 and v2.7. Version 2.7 was released on January 27, 2009 and consists of 439 classes and v2.9.2 was released on October 21, 2012 and consists of 458 classes. The test data set for inter-project validation was collected from another open source software named Sweet-Home-3D, which is an interior design application that helps you quickly draw the floor plan of your house, arrange furniture in it, and

view the results in 3D. We analyzed two versions of Sweet-Home-3D, which were v3.6 (349 classes) and v3.7 (350 classes). The details of the software used in the study are shown in Table 2. The analyzed versions of each software, its programming language (P/L used), the total LOC of each software, the total number of common classes extracted in the software (total classes), the number of common classes exhibiting changes (classes exhibiting change), and the number of classes that did not undergo changes (classes without change) of each software are all given in Table 2.

**Table 2.** Summary of data set used

| Name | Version 1 | Version 2 | P/L used | Total LOC | Total classes | Classes exhibiting change | Classes without change |
|---|---|---|---|---|---|---|---|
| Art-of-Illusion | 2.7 | 2.9.2 | Java | 88,444 | 434 | 131 | 303 |
| Sweet-Home-3D | 3.6 | 3.7 | Java | 74,762 | 348 | 15 | 333 |

### 3.3.1 Data collection procedure

In this work, we followed the steps given below to collect OO metrics and the changes between multiple versions of a software.

*Step I* (Metrics collection): The source code of both versions of the software was taken from http://sourceforge.net. The metrics listed in Table 1 were collected with the help of Understand software for the previous version of the software (i.e., AOI v2.7 and Sweet-Home-3D v3.6). The Understand software can provide the metric results for files, methods, etc. However, because we were assessing and predicting changes in only the OO classes, only the metrics for these classes were collected.

*Step II* (Preprocess versions): In this step, we extracted the common classes in both versions of the software. Changes in the common classes of both of the versions (current and previous) were taken into account. Zhou et al. [12] followed a similar approach in their work.

*Step III* (Compare classes): The common classes that were received after the preprocessing step were then compared line by line with the help of WinMerge software. Each common class of software was designated a binary status of 'change' or 'no change.' All of the classes were initially given a 'no change' status.

The standards that we followed for determining the status of the class are as follows:

- If there was any added, deleted, or modified line of source code in the current (newer) version of the class with respect to the previous version, it was marked as 'change.'
- The presence of any added, deleted, or modified comment in the class did not affect the status of the class. The status remained as 'no change.'
- The presence of any added or deleted blank line of source code in a class did not affect the status of the class. The status remained as 'no change.'
- If there was any change in the scope of class or class definition, it was marked as "change."
- If two lines were present in both versions of the class but were reordered then this did not affect the status of the class. The status remained as 'no change.'
- If there was any deletion or addition of unnecessary blank spaces or lines in a class, it did not affect the status of the class. The status remained as 'no change.'

Thus, all classes with a change in the lines of source code, class scope, or class definition were given a 'change' status and all other classes were given a 'no change' status. The status given to each class was the dependent variable of our study.

*Step IV* (Collection of data points): The change status from Step III and the metrics from Step I were combined to generate data points. Metric values and the change proneness (status) that corresponded to a class defined a data point. The number of data points in each software was the number of common classes found in both versions of the software. According to Table 2, the number of data points in AOI software is 434. In AOI software, 434 classes are present in both versions and 5 classes are absent in v2.9.2. A total of 131 classes exhibited a change in AOI (i.e., almost 30% of the total classes present in both versions). The data points collected from Sweet-Home-3D had 348 classes common in both the versions and only 1 class was absent in v3.7. Out of these 348 classes, only 15 classes were found changed. These details are summarized in Table 2.

### 3.3.2 Descriptive statistics

We analyzed the characteristics of the data sets that were used in this study on the basis of the descriptive statistics of various OO metrics given in Tables 3 and 4 for AOI and Sweet-Home-3D software, respectively. The table shows the minimum, maximum, mean, median, and standard deviation for each of the metrics analyzed in the study. The following observations were derived from these statistics:

1. The size of the class is measured in terms of lines of source code. It ranges from 4 to 3,053 LOC and 4 to 7,959 LOC for the AOI software and Sweet-Home-3D, respectively.
2. The mean value of the NOC metric (AOI, 0.463; Sweet-Home-3D, 0.433) and the DIT metric (AOI, 1.772; Sweet-Home-3D, 1.693) are low in the software systems that shows that number of children are very few and inheritance is least used in the systems under study. Similar results have been shown by others [4,5,24].
3. The LCOM measure (AOI, 100; Sweet-Home-3D, 100) is high for the software systems, which counts the number of classes with no common attribute usage.

**Table 3.** Descriptive statistics for AOI software

| Metric | Min | Max | Mean | Median | SD |
|---|---|---|---|---|---|
| NIB | 1 | 3 | 1.196 | 1 | 0.42 |
| CBO | 0 | 98 | 8.763 | 5.5 | 10.114 |
| NOC | 0 | 55 | 0.463 | 0 | 3.082 |
| NOA | 0 | 33 | 1.523 | 0 | 3.931 |
| RFC | 0 | 149 | 26.394 | 17.5 | 27.819 |
| NPRM | 0 | 23 | 1.581 | 0 | 3.05 |
| NPROM | 0 | 10 | 0.355 | 0 | 1.175 |
| NPM | 0 | 90 | 10.475 | 7 | 11.306 |
| SLOC | 4 | 3053 | 203.788 | 107 | 295.249 |
| DIT | 1 | 6 | 1.772 | 2 | 0.79 |
| MNL | 0 | 8 | 2.06 | 2 | 1.581 |
| LCOM | 0 | 100 | 50.696 | 59 | 31.243 |
| WMC | 0 | 554 | 35.267 | 19 | 54.295 |

**Table 4.** Descriptive statistics for Sweet-Home-3D software

| Metric | Min | Max | Mean | Median | SD |
|---|---|---|---|---|---|
| NIB | 1 | 5 | 1.473 | 1 | 0.667 |
| CBO | 0 | 70 | 5.994 | 4 | 8.319 |
| NOC | 0 | 37 | 0.433 | 0 | 2.547 |
| NOA | 0 | 45 | 0.908 | 0 | 3.542 |
| RFC | 1 | 279 | 17.662 | 10 | 25.119 |
| NPRM | 0 | 111 | 3.009 | 0 | 9.106 |
| NPROM | 0 | 52 | 0.616 | 0 | 3.126 |
| NPM | 0 | 84 | 7.891 | 4 | 11.524 |
| SLOC | 4 | 7959 | 214.573 | 65 | 557.09 |
| DIT | 1 | 4 | 1.693 | 2 | 0.678 |
| MNL | 0 | 9 | 1.854 | 1 | 1.765 |
| LCOM | 0 | 100 | 49.576 | 60 | 34.897 |
| WMC | 1 | 636 | 27.235 | 10 | 56.042 |

All abbreviations are listed in Table 1.

# 4. Experimental Framework

This section briefly describes all of the methods that we used in the study. It also gives a brief description of the model development techniques and the performance measures used to evaluate various change prediction models.

## 4.1 Statistical Model

Logistic regression is a method that is commonly used to predict a dependent variable from a set of independent variables (a detailed description is given in [2,25,26]). It is used when the dependent variable is dichotomous or binary. Univariate and multivariate regression, which are two types of logistic regression analysis, were used in this study. Univariate regression finds a connection between the dependent variable and each independent variable and it finds the significance of their association. On the other hand, multivariate regression builds a prediction model for the change proneness of classes. It analyzes the usefulness of metrics when the metrics are used in combination. Logistic regression results in a subset of significant metrics. To find this type of independent metrics subset, two stepwise selection methods are used, which are known as forward selection and backward elimination [26]. Forward selection analyzes one metric at the entry of each step. In backward elimination, all of the metrics in the model are selected at the initial step and are then removed one by one from the model until the stopping criteria is fulfilled. In this study, we used forward selection for statistical modeling.

Multivariate LR can be mathematically formulated as:

$$probab(X_1, X_2, \ldots\ldots X_n) = exp^{f(x)} / 1 + exp^{f(x)} \quad (1)$$

$$f(x) = C_0 + C_1X_1 + C_2X_2 + \ldots\ldots\ldots + C_nX_n \quad (2)$$

where $X_i$, $i = 1,2,\ldots,n$ are the independent variables. '*Probab*' is the probability of a class being change prone.

These are the following statistics for each significant metric in this particular model:

1. Odds ratio (OR): OR is calculated for each independent metric using $C_i$s. The formula for the odds ratio is OR = $exp(C_i)$. This is the ratio of probability that the event will occur to the probability of the non-occurrence of an event. The event is our case study of finding a change in the recent version of the class and non-occurrence is the failure to detect any change in the class [2,26].
2. Maximum-likelihood estimation (MLE) and coefficients ($C_i$s): The coefficients of a model are estimated by this statistical method. It is a likelihood function that records the probability of observing the set of dependent variables [25]. MLE finds the coefficients that make the log for the likelihood functions as large as possible. The larger the value, the better the impact of the independent variable on the predicted outcome variable [2,26].
3. The statistical significance (sig): Every coefficient has a significance level described by sig. The greater the value of the statistical significance, the lesser the impact of the independent OO metrics in prediction is. In our study we considered 0.05 to be the threshold for the significance level [2].

## 4.2 Machine Learning Methods

This section explains the machine learning methods that were used for conducting the analysis in our study. We used the default setting of the WEKA tool [27].

### 4.2.1 Correlation-based feature selection

Recognizing a representative set of features (i.e., OO metrics [independent variables]) is very important for building a classification model using machine learning methods [28]. A good feature set contains features that are highly correlated with the change in the class but that are not dependent on each other [29]. CFS helps to find the unwanted and noisy features that are correlated with other features. Hall [28] proved in his study that the "classification accuracy using a reduced feature set is equal to or better than the accuracy that is obtained when using a complete feature set." We utilized this method in our study for feature selection. The advantages of feature selection are reduction in dimensionality, improved predictive accuracy, and reduced execution time [28,29].

### 4.2.2 Random Forest (RF)

This is constructed from a collection of decision trees. Each of the trees is constructed by the random selection of a subset from a training dataset by using replacement. A decision tree is created from the random subset of the available variables. This helps us to create the best partitions for the dataset at each node. The final outcome is the one that is selected by the majority. Each decision tree gives its own vote for the result and the majority vote wins. RF is built using the unknown number of decision trees in which each decision tree has reached its full size. The advantages of RF are as follows: very little preprocessing of data and no need to select a variable to start building a model. RF itself selects the most useful variables [30,31].

### 4.2.3 AdaBoost

This stands for adaptive boosting. It can be used with other machine learning algorithms to improve the efficiency and performance of the model. It adapts to the error rates of the individual weak hypothesis. As the name suggests, it boosts the efficiency of a weak learning algorithm and converts it into a strong learning algorithm. The important concept of AdaBoost is that it maintains a distribution of weights over the training set. At the initial step, all of the weights are equal but with each successive iteration the weights of incorrect classified examples are increased so that the weak learning algorithm focuses more on the hard examples in the training set. This concept converts a weak algorithm into a strong one. The advantage of AdaBoost is that it is less susceptible to the over fitting problem than most other learning algorithms are [32].

### 4.2.4 Bagging

Bootstrap aggregating is a technique that repeatedly samples (with replacement) from a data set according to a uniform probability distribution [33]. It is also called bagging. The size of the bootstrap sample is same as the original data. As sampling is carried out with replacement, some instances may appear more than once in the same training set, while others may not be present at all. Every bootstrap sample contains 63% of the original training data on average. This is because each sample has the probability of $1 - (1 - 1/n)^n$ of being selected in each training set $D_i$. If n is sufficiently large enough the probability converges to $1 - 1/e = 0.632$. A test instance is assigned a class based on voting after training the $k$ classifiers [7].

### 4.2.5 Multilayer perceptron

Multilayer perceptron (MLP) are models that are used for computations that simulate biological neurons. MLP models complex relationships between inputs and outputs and is used to find patterns in data [34]. Multilayer feed-forward networks are those that have one input layer, one output layer, and one or more hidden layers that are used for complex mapping and to extract higher order statistics [34]. Back-propagation is the most commonly used learning paradigm to train multilayer feed-forward networks. It consists of two passes through that run the different layers of network. In the forward pass, a training input data set is applied and a set of outputs is produced as the actual response. In the backward pass, an error signal is generated, which is propagated backward through the network so that the synaptic weights can be readjusted to make the actual response closer to the desired response. The advantages of a neural network are its adaptive nature, non-linearity, parallel architecture, and fault tolerance.

### 4.2.6 Bayes Net

The Bayesian networks [35] are quite a powerful probabilistic representation and that is why they are often used for classification purposes. Unfortunately, perform poorly when learned in a standard way. Bayes Net is a graphical representation for probabilistic relationships among a set of random variables. Given a finite set $X$ $(X_1,X_2,X_3,\ldots,X_n)$ of discrete random variables where each variable $X_i$ may take values from a finite set, denoted by Val($X_i$). A Bayes Net is an annotated directed acyclic graph '$G$' that encodes

a joint probability distribution over $X$. In Bayes networks, the nodes of the graph correspond to the random variables $X_1, X_2,\ldots X_n$. The links of the graph correspond to the direct influence from one variable to the other. If there is a directed link from variable $X_i$ to variable $X_j$, variable $X_i$ will be a parent of variable $X_j$. Each node is annotated with a conditional probability distribution (CPD) that represents $p(X_i/P_a(X_i))$, where $P_a(X_i)$ denotes the parents of $X_i$ in $G$. The pair ($G$, $CPD$) encodes the joint distribution $p(X_i,\ldots X_n)$ [36].

### 4.2.7 Naïve Bayes

This is a well-established Bayesian method that was primarily formulated for performing classification tasks. Given its simplicity (i.e., the assumption that the independent variables are statistically independent), Naïve Bayes models are effective classification tools that are easy to use and interpret. Naïve Bayes is particularly appropriate when the dimensionality of the independent space (i.e., number of input variables) is high (a problem known as the curse of dimensionality). For the reasons given above, Naïve Bayes can often outperform other more sophisticated classification methods [37].

### 4.2.8 J48

J48 is an open implementation of the C4.8 algorithm by the WEKA tool in Java. This decision tree-based algorithm builds the tree in the same way as ID3 along with some improvements. Ros Quinlan developed this algorithm and it is now widely used for classification purposes. In this algorithm, first the base cases are checked and then for each attribute, the normalized information gain is found. The attribute that has the highest information gain is made the root node and this process is done recursively [38]. J48 is an evolved and refined offshoot of ID3 that accounts for unavailable values, continuous attribute value ranges, the pruning of decision trees, rule derivations, and so on. These properties make it more fruitful.

### 4.2.9 LogitBoost

LogitBoost is one of the boosting techniques that were proposed by Friedman et al. [39]. Boosting helps to change a weak learning algorithm into a strong learning algorithm. This is done by assigning equal weights to all the training examples and then increasing the weights of incorrectly classified examples. In this way, a weak learner is forced to focus on the hard examples. In other words, boosting is the process of applying a classification algorithm to the training instances, reweighting them again and again, and then taking a majority vote on the number of classifiers thus produced. The LogitBoost algorithm takes the AdaBoost algorithm as an additive model and applies the cost function of logistic regression [39]. LogitBoost is suitable for problems involving two class situations.

### 4.2.10 NNge

NNge is the nearest-neighbor-like algorithm that uses non-nested generalized exemplars (which are hyper-rectangles that can be viewed as ‘if-then’ rules). It is an algorithm that was developed to determine the solution to the famous travelling salesman problem. It classifies a new example ‘a’ by finding

the training example (a,b) that is nearest to 'a' according to the Euclidean distance. The nearest neighbor algorithm follows the greedy approach to find the shortest distance. It is easy to implement and executes quickly. The disadvantage of the NNge algorithm is that it may not find a feasible route at all, even when one exists.

## 4.3 Model Development Techniques

This study evaluates the following two types of change prediction models: 1) models developed using ten-fold cross validation where the training, as well as the validation set belongs to the same software and 2) models developed using inter-project validation. We briefly explain the two procedures as laid out below.

1. Ten-Fold Cross Validation: This procedure involves the random division of data into ten parts. Each time, nine parts are used as a training set and the left out part is used as a test set [40]. Therefore, we obtain the change proneness of all ten parts. Applying a prediction model to different data sets gives a better view of its accuracy. Hence, we developed models using the ten-fold cross validation procedure. However, while using this procedure we used the data points of the same software while performing model training or during model validation. This study analyzes the ten-fold cross validation results of both the AOI data set and the Sweet-Home-3D data set.
2. Inter-Project Validation: A model would be more effective and useful if it can give good results on a validation set that is different (belongs to a different data set) from the training set. This procedure involves using data points of a specific type of software for model training and validating the model on the data points of a different type of software. Thus, inter project validation can assess the possibility of using developed models on different software data sets, which can lead to a lot of saving of resources. In order to perform inter-project validation, this study uses the AOI software as the training data set and the Sweet-Home-3D data set as the test set. A good inter-project model supports the reusability of the already established models on different data sets effectively.

## 4.4 Performance Measures

We used the measures described below to evaluate the performance of the change proneness prediction models that we developed in our study.

1. *Sensitivity and Specificity:* Sensitivity and specificity represents the correctness of these models. The percentage of classes that have been correctly predicted to be prone to change from among all the actual change prone classes is known as the sensitivity of the model. The opposite of sensitivity (i.e., the percentage of classes correctly predicted to be not change prone) is called the specificity of the model. The values of both the sensitivity and specificity should be high, in order to predict change-prone and non-change prone classes correctly.
2. *ROC Analysis:* Outputs of prediction models are evaluated using ROC analysis. The ROC curve is plotted between 1-specificity on the x-axis and sensitivity on the y-axis, which effectively predicts the quality of predicted models [3,41]. We selected many cutoff points between 0 and 1 while constructing ROC curves and calculated the sensitivity and specificity on those points. The optimal choice of the cutoff point (that maximizes both sensitivity and specificity) can be

selected from the curve [3,41]. AUC is a combined measure of sensitivity and specificity [3]. The AUC, which is computed by using ROC analysis, computes the accuracy of a model.

# 5. Analysis and Results

This section describes the type of analysis that is performed to find the relationship between OO metrics and the change proneness of classes. We employed a statistical method, which is the multivariate logistic regression, to find the combined effect of OO metrics on the change proneness attribute of a class. Apart from this method, we also employed nine machine learning methods to analyze the relationship between OO metrics and the change proneness of classes. The methods we used are Bayes Net, Naive Bayes, MLP, Bagging, AdaBoost, LogitBoost, J48, NNge, and RF.

## 5.1 Attribute Selection Results

In order to construct an effective prediction model, a subset of metrics from Table 1 was selected by applying the CFS technique. The selected subset of metrics for the AOI data set was comprised of CBO, NPRM, SLOC, LCOM, and WMC metrics. CFS is only applied for model development using the machine learning methods. For the statistical method (LR), stepwise forward selection was used, which selected the CBO, RFC, NPRM, and DIT metrics for AOI software. Table 5 shows the results of the metrics that were selected using the CFS or the forward stepwise selection on the AOI data set. The set of metrics selected after the CFS technique was applied to the Sweet-Home-3D data set were the NPRM, SLOC, and WMC metric. The metrics that were selected by using forward stepwise selection on the Sweet-Home-3D data sets was NPRM and NPROM (Table 5).

**Table 5.** Metrics selected by CFS and stepwise forward selection LR method

| Software name | Metric selection method | Metrics selected |
|---|---|---|
| Art-of-Illusion | CFS | CBO, NPRM, SLOC, LCOM, WMC |
| | Forward LR | CBO, RFC, NPRM, DIT |
| Sweet-Home-3D | CFS | NPRM, SLOC, WMC |
| | Forward LR | NPRM, NPROM |

All abbreviations are listed in Table 1.

## 5.2 Result Analysis for Art-of-Illusion

This section describes the validation results of the various models that were developed using the AOI software. Table 6 shows the coefficient (B), standard error (SE), statistical significance (sig), and odds ratio (OR) for the metrics included in the multivariate LR prediction model for the AOI data set. Table 7 shows the ten-fold cross validation results on the system that were studied (i.e., AOI). The sensitivity and specificity values for the model developed using the LR method were 63.4% each with a cutoff point of 0.235. The AUC for LR was 0.719, which is much less than the AUC achieved by the model developed using the Bagging method, which is 0.803. The sensitivity and specificity values for the Bagging model are also much higher at 71% and 71.3%, respectively. The cutoff point for Bagging was 0.291. The

RF method gives some competitive results with an AUC value of 0.784 and the sensitivity and specificity values of 75.6% and 67%, respectively. The cutoff point for the RF method was 0.25. The Bayes Net and LogitBoost methods also gave good results. Detailed results of the validation for all the change proneness prediction models using AOI software are stated in Table 7.

**Table 6.** Result of multivariate LR analysis of Art-of-Illusion software

| Metric | B | SE | Sig. | OR |
|---|---|---|---|---|
| CBO | 0.078 | 0.021 | 0.000 | 1.081 |
| RFC | 0.024 | 0.008 | 0.001 | 1.025 |
| NPRM | 0.242 | 0.052 | 0.000 | 1.273 |
| DIT | -0.878 | 0.234 | 0.000 | 0.416 |
| Constant | -1.044 | 0.312 | 0.001 | 0.352 |

**Table 7.** Art-of-Illusion validation results

| Method | Specificity | Sensitivity | Cutoff point | AUC |
|---|---|---|---|---|
| LR | 63.4 | 63.4 | 0.235 | 0.719 |
| Bayes Net | 68.0 | 67.9 | 0.252 | 0.765 |
| Naïve Bayes | 67.3 | 66.4 | 0.012 | 0.712 |
| MLP | 67.0 | 67.2 | 0.251 | 0.730 |
| Bagging | 71.3 | 71.0 | 0.291 | 0.803 |
| AdaBoost | 67.0 | 66.4 | 0.386 | 0.752 |
| LogitBoost | 69.3 | 68.7 | 0.298 | 0.782 |
| NNge | 81.8 | 55.0 | 0.500 | 0.684 |
| J48 | 57.4 | 66.4 | 0.245 | 0.684 |
| RF | 67.0 | 75.6 | 0.250 | 0.784 |

## 5.3 Result Analysis for Sweet-Home-3D

The results of multivariate LR analysis for the Sweet-Home-3D data set are shown in Table 8. Ten-fold cross validation results for the prediction models of the Sweet-Home-3D, along with the forward LR model, are provided in detail, as shown in Table 9. According to Table 9, Naïve Bayes provides the best prediction results with an AUC value of 0.879 and high specificity (79.6%) and sensitivity (80%) values. Other machine learning models also performed well in comparison to the statistical model. The performances by J48 and NNge were the lowest among other models with AUC values at 0.414 and 0.591, respectively. Sensitivity and specificity for J48 and NNge were also low in the case of the Sweet-Home-3D software system. The performance of the Naïve Bayes method was slightly better than the MLP method with a high AUC value (0.846) and a very good sensitivity value (80%).

**Table 8.** Result of multivariate LR analysis of Sweet-Home-3D software

| Metric | B | SE | Sig. | OR |
|---|---|---|---|---|
| NPRM | 0.289 | 0.051 | 0.000 | 1.336 |
| NPROM | -27.262 | 1695.25 | 0.987 | 0.000 |
| Constant | -4.394 | 0.506 | 0.000 | 0.012 |

**Table 9.** Sweet-Home-3D validation results

| Method | Specificity | Sensitivity | Cutoff point | AUC |
|---|---|---|---|---|
| LR | 72.7 | 73.3 | 0.018 | 0.780 |
| Bayes Net | 81.7 | 80.0 | 0.005 | 0.819 |
| Naïve Bayes | 79.6 | 80.0 | 0.011 | 0.879 |
| MLP | 79.6 | 80.0 | 0.030 | 0.846 |
| Bagging | 75.7 | 80.0 | 0.037 | 0.796 |
| AdaBoost | 77.2 | 80.0 | 0.013 | 0.821 |
| LogitBoost | 68.2 | 73.3 | 0.013 | 0.802 |
| NNge | 98.2 | 20.0 | 0.500 | 0.591 |
| J48 | 49.5 | 33.3 | 0.044 | 0.414 |
| RF | 89.2 | 53.3 | 0.050 | 0.730 |

**Table 10.** t-test to check statistical significance

| Method | Art-of-Illusion | Sweet-Home-3D |
|---|---|---|
| Bagging | 0 | 0 |
| AdaBoost | + | 0 |
| LR | + | 0 |
| MLP | + | 0 |
| Bayes Net | 0 | 0 |
| Naïve Bayes | + | 0 |
| Random Forest | 0 | 0 |
| J48 | + | + |
| NNge | + | + |

## 5.4 Hypothesis Validation

We compared the percentage of correct statistics of one of the machine learning methods (LogitBoost) with the other nine methods. This was done in order to check whether the different methods used in the study were statistically significantly different from each other for developing change prediction models. For this, we used the t-test with a significance level of 0.05. We used the open source tool, WEKA, to perform this test.

The number '0' indicates that this particular method is not statistically different from the baseline method (LogitBoost) at the specified significance level of 0.05. Whereas, the symbol '+' indicates that the method is statistically superior to the baseline method (LogitBoost) at the specified significance level of 0.05. We performed this test for both the AOI and Sweet-Home-3D software. We can observe from Table 10 that for AOI, that Adaboost, LR, MLP, Naïve Bayes, J48, and NNge are statistically better or superior than LogitBoost (rejection of the null hypothesis). Whereas, for Sweet-Home-3D, we can observe that the majority of the methods are statistically the same as the baseline method of LogitBoost (acceptance of the null hypothesis). Thus, overall we can say that the results are either statistically superior or the same as the baseline method and that no method has shown statistically worse performance than the baseline method.

## 5.5 Inter-Project Validation of the Prediction Model Using Sweet-Home-3D

In order to perform inter project validation, we developed a model using AOI software as the training set and Sweet-Home-3D software as the validation set. Inter-project models were developed using only machine learning methods. The results of the developed models are shown in Table 11. According to the table, the results of all the prediction models are very good. The Bayes Net model outperformed other machine learning models with a maximum AUC value of 0.913. The sensitivity and specificity values for the model developed by using the Bayes Net method were 79.6% and 80%, respectively, with a cutoff 0.664. LogitBoost and Naïve Bayes models also gave competitive results with AUC values of 0.882 and 0.901, respectively. The sensitivity and specificity values were also high for these models. Other prediction models that were developed using machine learning methods for change prone classes also demonstrated high efficiency, as indicated in Table 11.

**Table 11.** Inter-project validation results with Sweet-Home-3D test set

| Method | Specificity | Sensitivity | Cutoff point | AUC |
|---|---|---|---|---|
| Bayes Net | 79.6 | 80.0 | 0.664 | 0.913 |
| Naïve Bayes | 83.8 | 80.0 | 0.250 | 0.901 |
| MLP | 79.6 | 80.0 | 0.355 | 0.873 |
| Bagging | 79.9 | 80.0 | 0.419 | 0.847 |
| AdaBoost | 68.8 | 73.3 | 0.411 | 0.844 |
| LogitBoost | 80.2 | 80.0 | 0.417 | 0.882 |
| NNge | 83.8 | 80.0 | 0.500 | 0.819 |
| J48 | 93.7 | 66.7 | 0.545 | 0.799 |
| RF | 73.6 | 73.3 | 0.350 | 0.814 |

## 5.6 Discussion of Results

In this section, we compare the performance of the models predicted for both of the projects (AOI and Sweet-Home-3D). Various machine learning and statistical models were predicted and their performance was measured using various performance measures, such as sensitivity, specificity, and area under the ROC curve. As per the definitions of these measures (provided in Section 5), high values of these measures are desired. The higher the values, the better the model is in terms of accuracy.

The ten-fold cross validation results for the AOI data set suggests Bagging as being the best method from among all of the machine learning and statistical methods with an AUC value of 0.803. However, the validation results for Sweet-Home-3D have shown that the model that was developed using Naïve Bayes as being the best prediction model with high values of AUC (0.879) and sensitivity (80%). Thus, we can say that the validation results vary as the dataset changes and are not alike for different datasets. However, if we have a dataset with similar characteristics, then the results may be similar (i.e., we may get the same machine learning method as being the best one). The other machine learning methods viz. Bayes Net, RF, and LogitBoost also gave comparable performances.

These models can be used for predicting the change proneness of classes. The prediction of change prone classes in the early phases of software development will help us in effectively planning testing resources. Change prone classes should be rigorously tested by allocating more resources to them, as a

change prone class requires a greater effort to be made during the maintenance phase. A major source of change in change prone classes could be defects and testing these classes meticulously would decrease the probability of a defect. Thus, the early prediction of such classes reduces our efforts during testing and maintenance phases.

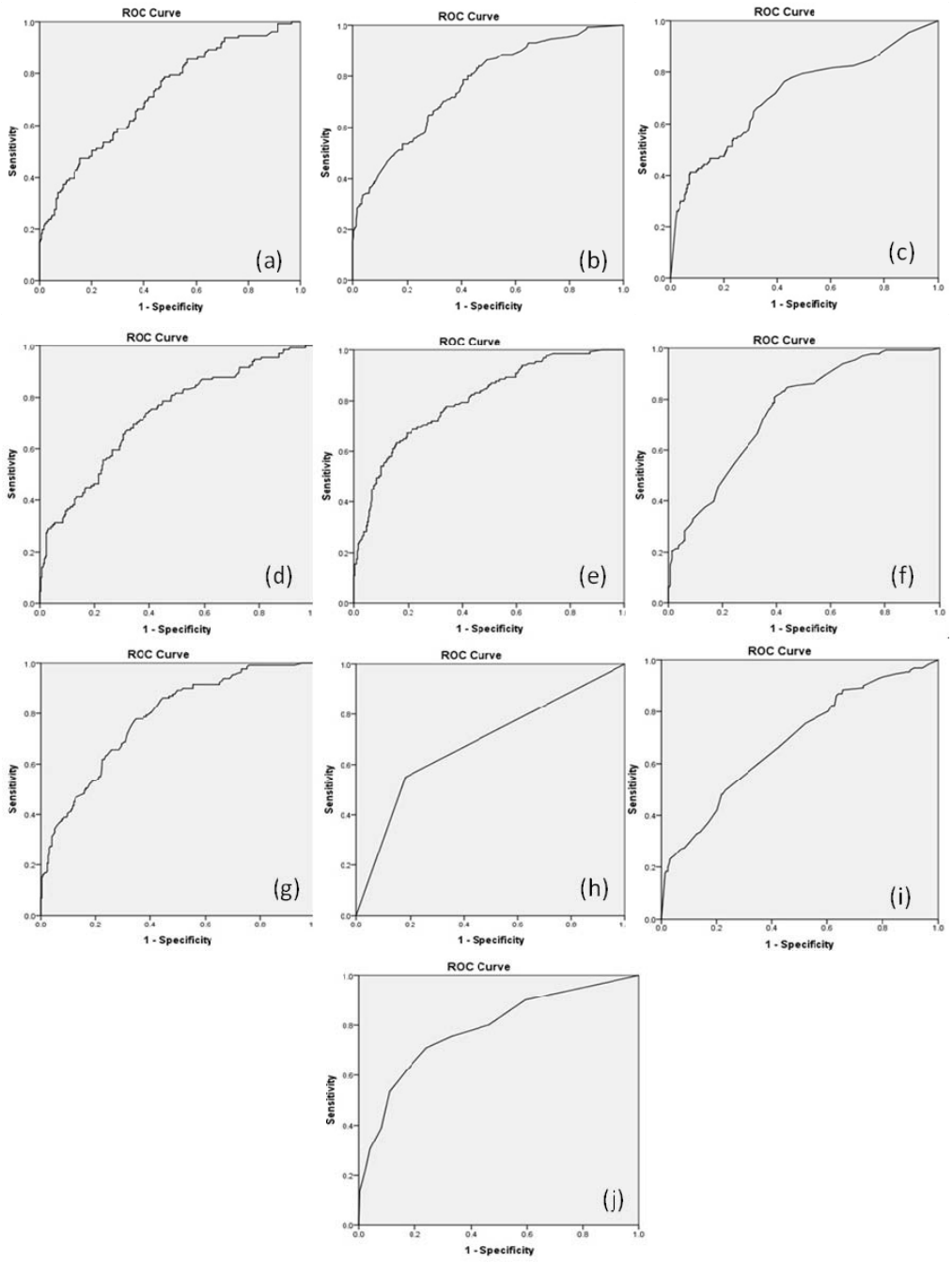


**Fig. 2.** Validation curves of AOI: (a) Forward LR, (b) Bayes Net, (c) Naïve Bayes, (d) MLP, (e) Bagging, (f) AdaBoost, (g) LogitBoost, (h) NNge, (i) J48, and (j) RF.

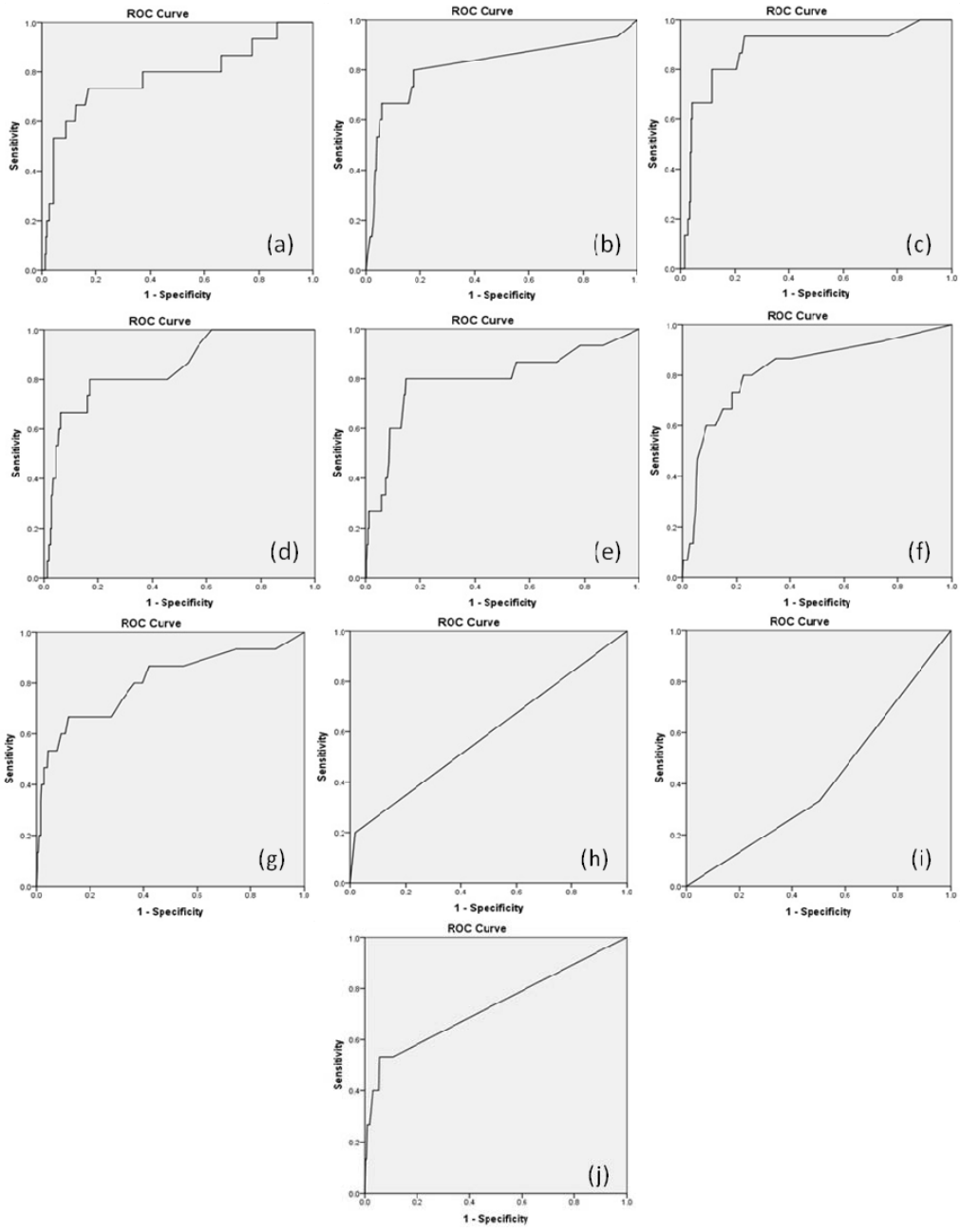


**Fig. 3.** Validation curves of Sweet-Home-3D: (a) Forward LR, (b) Bayes Net, (c) Naïve Bayes, (d) MLP, (e) Bagging, (f) AdaBoost, (g) LogitBoost, (h) NNge, (i) J48, and (j) RF.

In this section, we compare the performance of the models predicted for both of the projects (AOI and Sweet-Home-3D). Various machine learning and statistical models were predicted and their performance was measured using various performance measures, such as sensitivity, specificity, and area under the ROC curve. As per the definitions of these measures (provided in Section 5), high values of these measures are desired. The higher the values, the better the model is in terms of accuracy.

The ten-fold cross validation results for the AOI data set suggests Bagging as being the best method from among all of the machine learning and statistical methods with an AUC value of 0.803. However, the validation results for Sweet-Home-3D have shown that the model that was developed using Naïve Bayes as being the best prediction model with high values of AUC (0.879) and sensitivity (80%). Thus, we can say that the validation results vary as the dataset changes and are not alike for different datasets. However, if we have a dataset with similar characteristics, then the results may be similar (i.e., we may get the same machine learning method as being the best one). The other machine learning methods viz. Bayes Net, RF, and LogitBoost also gave comparable performances.

These models can be used for predicting the change proneness of classes. The prediction of change prone classes in the early phases of software development will help us in effectively planning testing resources. Change prone classes should be rigorously tested by allocating more resources to them, as a change prone class requires a greater effort to be made during the maintenance phase. A major source of change in change prone classes could be defects and testing these classes meticulously would decrease the probability of a defect. Thus, the early prediction of such classes reduces our efforts during testing and maintenance phases.

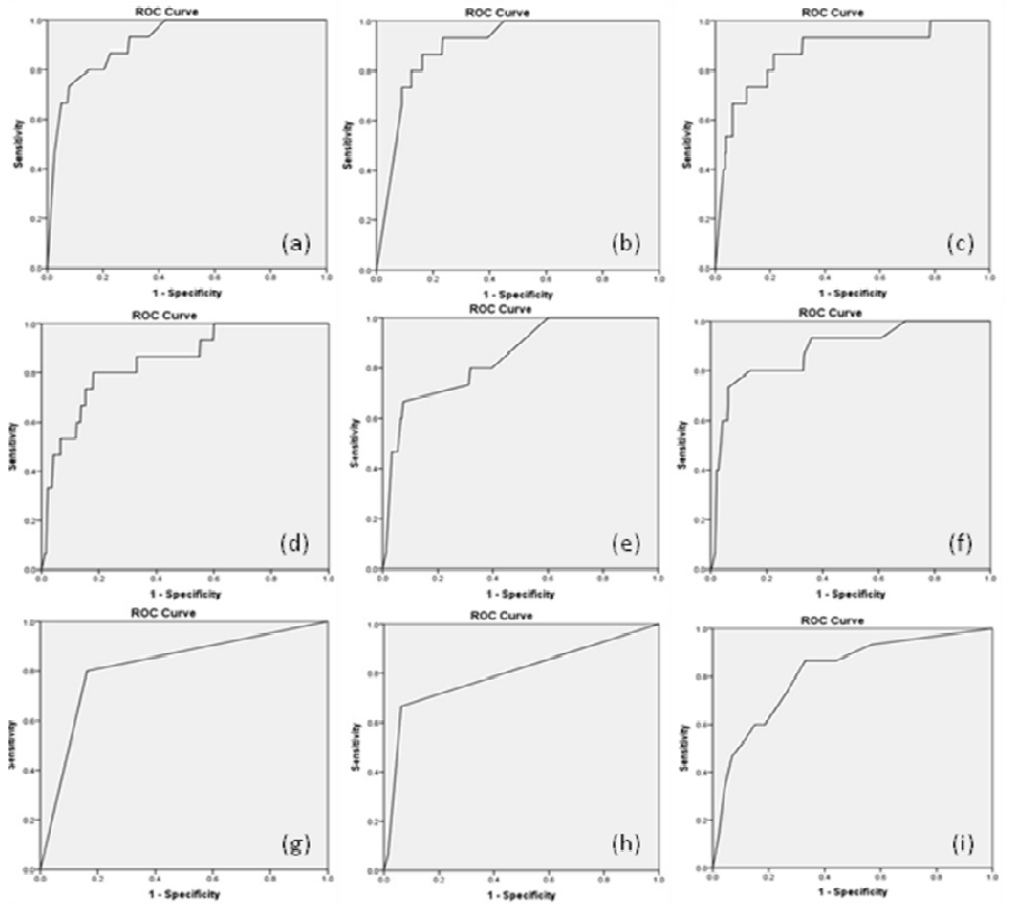


**Fig. 4.** Inter-project validation curves: (a) Bayes Net, (b) Naïve Bayes, (c) MLP, (d) Bagging, (e) AdaBoost, (f) LogitBoost, (g) NNge, (h) J48, and (i) RF.

Fig. 2 shows the ROC curves of prediction models for AOI, Fig. 3 shows the ROC curves for Sweet-Home-3D, and Fig. 4 shows the ROC curves for inter-project validation. Thus, we can conclude that the results of machine learning methods are competitive and comparable to the statistical method (i.e., LR). Another conclusion from our study is that NPRM, SLOC, and WMC are good indicators of change proneness as they are selected in the reduced feature sets of both data sets after the CFS technique is applied. Another important result concluded from our analysis of inter-project validation models is that we can use already constructed models for predicting the change prone classes of other software from the same domain.

# 6. Threats to Validity

This section discusses the three areas of concern related to this study, which are as follows: construct validity, external validity and internal validity, which are common to all empirical studies.

## 6.1 Construct Validity

Construct validity refers to the extent to which the dependent and independent variables accurately measure the concept that they are supposed to measure [1]. Corrective, adaptive, and perfective or preventive are different classes of change in software [1,41]. The types of change that a class goes through are not segregated in our study. A similar approach was followed in [1,43]. The metrics included in our study cover concepts like cohesion, coupling, inheritance, etc. These metrics were already validated in previous studies [4,44,45] (i.e., they are accurate measure of the concepts that they represent). The change proneness or dependent variable is calculated manually with the help of WinMerge software. Thus, the threat of collecting an incorrect dependent variable is removed.

## 6.2 Internal Validity

Zhou et al. [12] stated that "Internal validity is the degree to which a conclusion can be drawn about the causal effect of independent variables on the dependent variables." We empirically determined the relation between independent variables and the change proneness attribute of a class. However, we did not demonstrate the causal relationship of various attributes on the dependent variable. This can only be evaluated by varying a particular measure (like coupling) and keeping the functionality constant [46]. This type of controlled experiment is difficult to carry out in practice and this threat exists in various studies [46]. Our main focus was not to analyze the causal relationship between OO metrics and change proneness.

## 6.3 External Validity

This refers to the degree of generalization of the results. This can be achieved by analyzing and comparing the results that were achieved on a vast number of data sets [1]. We only used two open source software in our study. Thus, the results cannot be generalized to all software systems. However, previous studies in this area with different projects in different languages have proved the relationship be-

tween OO metrics and the change proneness attribute of a class. Furthermore, we have successfully performed the inter-project validation of prediction models using other software from the same domain (i.e., 3D modeling). The results conclude that these models can be used for predicting the change prone classes of the software systems from the same domain, but we cannot generalize these models for software systems from other domains. We can eliminate this threat by carrying out replicated studies in this area with different data sets belonging to different domains.

# 7. Application of the Work

This study has developed various models for predicting the change prone classes during the early phases of the software development life cycle. The prediction of change prone classes would help developers in judiciously allocating the resources and pay focused attention in terms of cost, time and effort on these classes. This would help in delivering better quality software products as effective testing would decrease the number of defects in a software and would lead to more satisfied customers.

Change prediction models were developed using various OO metrics that are available at the design phase. From a suite of 13 OO metrics, the univariate analysis gave CBO and NPRM metrics as significant predictors of change proneness for both of the projects (AOI and Sweet-Home-3D). Using the CFS technique, NPRM, SLOC, and WMC were found to be significant indicators of change proneness. Researchers and practitioners can use these significant predictors in the earlier phases of software development to measure the quality of the systems. Thus, closely monitoring these metrics would give developers an idea of the ideal values of these metrics. Thus, we can alter the design measurements if we find the design to be inappropriate. These design measurements can be used throughout different organizations as quality benchmarks to assess and compare different industrial projects. Furthermore, within this same project, we are able to check the design of the upcoming release and suggest any alternations, if required. This helps in checking the quality of the new versions, which are released on a regular basis. In order to draw stronger conclusions, more replicated studies are required.

# 8. Conclusions

The aim of the study was to analyze the relationship between OO metrics (independent variables) and change proneness of a class. In order to explore this relationship, we empirically validated the results of change prediction models on two open source software, AOI and Sweet-Home-3D. The second objective of the study was to analyze, compare and evaluate the performance of logistic regression and machine learning methods in predicting change prone classes of a software. The study evaluates the performances of a number of machine learning and statistical change prediction models developed on the two software data sets of the study using ROC analysis. This study also performed a statistical test (t-test) to statistically compare the models developed using different methods on both the data sets. Another objective of the study was to ascertain whether inter-project change prediction models can be successfully used for predicting the change prone classes of a different software. To achieve this objective, we performed inter-project validation by using the prediction model developed with AOI software as the training set and Sweet-Home-3D software as the test set. Since both the software were from the

same domain, our results would help the industry and academia to analyze whether these prediction models can be used to predict change prone classes for the software data sets belonging to a similar domain. Thus, the contribution of this work can be summarized as follows: First, we performed analysis on both open source software data sets taking change in classes in account for predicting change proneness. Second, apart from statistical method, we applied nine different machine learning methods to construct prediction models for change prone classes and evaluated their performances. Third, we analyze and assess the applicability of inter-project models for change prediction. The key findings and recommendations of the study are summarized as follows:

1. NPRM, SLOC and WMC metrics are significant indicators of change proneness in both the data sets as they were selected after applying the feature subset selection technique, CFS. CBO and NPRM metrics were found significant using univariate LR analysis. Thus, the values of these metrics (NPRM, SLOC, WMC, CBO, and NPRM) should be closely monitored during development and maintenance of a product as they are closely associated with change in a class. Developers can formulate quality benchmarks stating appropriate values for these metrics so that the quality of the software product can be monitored regularly. Corrective actions should be taken if at any point the product is not in accordance with the quality benchmarks.
2. For both the software data sets used in the study, the models developed using the Bagging, RF, Bayes Net, LogitBoost and MLP methods outperformed the model developed using the LR method. These models yielded good AUC values. This concludes that machine learning methods are competitive and comparable with statistical methods for predicting change prone classes in a software. Thus, software developers can use machine learning methods for developing effective change prediction models.
3. The results of validation for 'AIO' project showed the best performance by Naïve Bayes method, whereas the results of validation for 'Sweet-Home-3D' showed the best performance by Bagging method. Thus we conclude that it is not necessary that the results of validation for different projects will be similar. Researchers and developers should take into account the characteristics of the data sets as well as the properties of the methods before choosing them to develop change prediction model for a particular data set.
4. The AUC of machine learning based prediction models range from 0.799 to 0.913 when we performed inter-project validation on Sweet-Home-3D data set. Hence, inter-project validation using Sweet-Home-3D as test data set, concludes that prediction models can successfully predict the change prone classes for other software data set from the same domain. Inter-project models save a lot of effort and time as we do not need to collect training data specific to a project. Such models are useful when training data specific to a project may not be previously available or is very expensive to compute.
5. The results of this study (effective change prediction models) can be used by researchers and practitioners in the early phases of software development life cycle to reduce the maintenance effort and costs. It can also be used for effectively planning testing resources. This is possible as change prone classes need more attention and resources while maintenance and testing. These classes are changed due to existing errors or a new requirement. Constantly monitoring and focusing resources on these classes would help in identifying probable errors and changes in the early phases of software development. Thus, developers can correct errors and may perform corrective design measures to reduce the changes and errors in the later stages of software development. Thus, there is a reduction in maintenance costs and effective allocation of test resources.

This work is limited to a medium-sized object oriented open source project. So, it may not work well in case of large software systems. We plan to replicate our study to predict models based on other highly popular machine learning methods like ant colony optimization and particle swarm optimization. We also plan to study the economical benefit of change proneness models.

## References


[1] H. Lu, Y. Zhou, B. Xu, H. Leung, and L. Chen, "The ability of object-oriented metrics to predict change-proneness: a meta-analysis," *Empirical Software Engineering*, vol. 17, no. 3, pp. 200-242, 2012.

[2] K. K. Aggarwal, Y. Singh, A. Kaur, and R. Malhotra, "Empirical analysis for investigating the effect of object-oriented metrics on fault proneness: a replicated case study," *Software Process: Improvement and Practice*, vol. 14, no. 1, pp. 39-62, 2009.

[3] Y. Singh, A. Kaur, and R. Malhotra, "Empirical validation of object-oriented metrics for predicting fault proneness," *Software Quality Journal*, vol. 18, no. 1, pp. 3-35, 2010.

[4] L. C. Briand, J. Wust, J. W. Daly, and D. Victor Porter, "Exploring the relationships between design measures and software quality in object-oriented systems," *Journal of Systems and Software*, vol. 51, no. 3, pp. 245-273, 2000.

[5] M. Cartwright and M. Shepperd, "An empirical investigation of an object-oriented software system," *IEEE Transactions on Software Engineering*, vol. 26, no. 8, pp. 786-796, 2000.

[6] M. English, C. Exton, I. Rigon, and B. Clearyp, "Fault detection and prediction in an open source software project," in *Proceedings of the 5th International Conference on Predictor Models in Software Engineering (PROMISE2009)*, Vancouver, Canada, 2009.

[7] R. Malhotra and A. Jain, "Software effort prediction using statistical and machine learning method," *International Journal of Advanced Computer Science and Applications*, vol. 2, no. 1, pp. 145-152, 2011.

[8] A. Han, S. Jeon, D. Bae, and J. Hong, "Behavioral dependency measurement for change proneness prediction in UML 2.0 design models," in *Proceedings of the 32nd Annual IEEE International Conference on Computer Software and Applications (COMPSAC2008)*, Turku, Finland, 2008, pp. 76-83.

[9] M. Ajrnal Chaumum, H. Kabaili, R. K. Keller, and F. Lustman, "A change impact model for changeability assessment in object oriented software systems," in *Proceedings of the 3rd European Conference on Software Maintenance and Reengineering*, Amsterdam, 1999, pp. 130-138.

[10] N. Tsantalis, A. Chatzigeorgiou, and G. Stephanides, "Predicting the probability of change in object oriented systems," *IEEE Transactions on Software Engineering*, vol. 31, no. 7, pp. 601-614, 2005.

[11] A. R. Sharafat and L. Tavildari, "A probabilistic approach to predict change in object-oriented software systems," in *Proceedings of the 11th European Conference on Software Maintenance and Reengineering (CSMR2007)*, Amsterdam, 2007, pp. 27-38.

[12] Y. Zhou, H. Leung, and B. Xu, "Examining the potentially confounding effect of class size on the associations between object-oriented metrics and change-proneness," *IEEE Transactions on Software Engineering*, vol. 35, no. 5, pp. 607-623, 2009.

[13] L. Malhotra and A. J. Bansal, "Prediction of change prone classes using machine learning and statistical techniques," in *Advanced Research and Trends in New Technologies, Software, Human-Computer Interaction and Communicability*. Hershey, PA: IGI Global, 2014, pp. 193-202.

[14] Understand your code official website [Online]. Available: http://www.scitools.com.

[15] Understand your code - About SciTools [Online]. Available: http://www.scitools.com/about/index.php.

[16] Understand your code - Metrics [Online]. Available: https://scitools.com/feature/metrics/.

[17] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476-493, 1994.

[18] K. K. Aggarwal, Y. Singh, A. Kaur, and R. Malhotra, "Empirical study of object-oriented metrics," *Journal of Object Technology*, vol. 5, no. 8, pp. 149-173, 2006.

[19] B. Henderson-Sellers, *Object-Oriented Metrics: Measures of Complexity*. Englewood Cliffs, NJ: Prentice Hall, 1996.

[20] M. Lorenz and J. Kidd, *Object-oriented Software Metrics: A Practical Guide*. Englewood Cliffs, NJ: Prentice Hall, 1994.

[21] Understand your code - County metrics [Online]. Available: https://scitools.com/support/metrics_list.

[22] Understand your code - Complexity metrics [Online]. Available: https://scitools.com/support/metrics_list.

[23] Understand your code - Object oriented metrics [Online]. Available: https://scitools.com/support/ metrics_list.

[24] S. R. Chidamber, D. P. Darcy, and C. F. Kemerer, "Managerial use of metrics for object-oriented software: an exploratory analysis," *IEEE Transactions on Software Engineering*, vol. 24, no. 8, pp. 629-639, 1998.

[25] V. R. Basili, L. C. Briand and W. L. Melo, "A validation of object-oriented design metrics as quality indicators," *IEEE Transactions on Software Engineering*, vol. 22, no. 10, pp. 751-761, 1996.

[26] D. W. Hosmer and S. Lemeshow, *Applied Logistic Regression*. New York, NY: Wiley, 1989.

[27] Weka 3: Data Mining Software in Java [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/.

[28] M. A. Hall, "Correlation based feature selection for discrete and numeric class machine learning," in *Proceedings of the 17th International Conference on Machine Learning (ICML2000)*, Stanford, CA, 2000, pp. 359-366.

[29] K. Michalak and H. Kwasnicka, "Correlation based feature selection strategy in neural classification," in *Proceedings of the 6th International Conference on Intelligent Systems Design and Applications (ISDA2006)*, Jinan, China, 2006, pp.741-746.

[30] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. Burlington, MA: Morgan Kaufmann, 2011.

[31] Y. Freund, R. E. Schapire, and N. Abe, "A short introduction to boosting," *Journal of Japanese Society for Artificial Intelligence*, vol. 14, no. 5, pp. 771-780, 1999.

[32] AdaBoost [Online]: Available: http://en.wikipedia.org/wiki/AdaBoost.

[33] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123-140, 1996.

[34] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Delhi: Pearson education, 1999.

[35] J. Pearl, Bayesian Networks, 1988 [Online]: Available: http://ftp.cs.ucla.edu/pub/stat_ser/R246.pdf.

[36] Bayes Nets [Online]. Available: http://www.bayesnets.com/.

[37] K. P. Murphy, "Naive Bayes classifiers," *Technical Report*, University of British Columbia, Canada, 2006.

[38] C4.5 algorithm [Online]. Available: http://en.wikipedia.org/wiki/C4.5_algorithm.

[39] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *The Annals of Statistics*, vol. 28, no. 2, pp. 337-407, 2000.

[40] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *Journal of the Royal Statistical Society Series B: Methodological*, vol. 36, no 2, pp. 111-147, 1974.

[41] K. El-Emam, S. Benlarbi, N. Goel, and SN. Rai, "A validation of object-oriented metrics," *Technical Report ERB-1063*, National Research Council of Canada, 1999.

[42] K. K. Aggarwal and Y. Singh, *Software Engineering*, 2nd ed. New Delhi: New Age International Publishers, 2007.

[43] A. G. Koru and J. Tian, "Comparing high-change modules and modules with the highest measurement values in two large-scale open-source products," *IEEE Transactions on Software Engineering*, vol. 31, no. 8, pp. 625-642, 2005.

[44] L. C. Briand, J. Daly, and J. Wust, "A unified framework for cohesion measurement in object-oriented systems," *Empirical Software Engineering*, vol. 3, no. 1, pp. 65-117, 1998.

[45] L. C. Briand, J. Daly, and J. Wust, "A unified framework for coupling measurement in object-oriented systems," *IEEE Transactions on Software Engineering*, vol. 25, no. 1, pp. 91-121, 1999.

[46] L. C. Briand, J. Wust, and H. Lounis, "Replicated case studies for investigating quality factors in object-oriented designs," *Empirical Software Engineering*, vol. 6, no. 1, pp. 11-58, 2001.

**Ruchika Malhotra**

She is an Assistant Professor at the Department of Software Engineering, Delhi Technological University (formerly Delhi College of Engineering), Delhi, India. She has been awarded the prestigious UGC Raman postdoctoral fellowship by the Indian government and is currently pursuing postdoctoral research from Department of Computer and Information Science, Purdue University, Indianapolis, USA. She was an assistant professor at the University School of Information Technology, Guru Gobind Singh Indraprastha University, Delhi, India. She has received her master's and doctorate degree in software engineering from the University School of Information Technology, Guru Gobind Singh Indraprastha University, Delhi, India. She has received IBM Faculty Award 2013. She is executive editor of Software Engineering: An International Journal. She is co-author of a book on Object Oriented Software Engineering published by PHI learning. Her research interests are in software testing, improving software quality and the definition and validation of software metrics. She has published more than 80 publications in international journals and conferences.

**Ravi Jangra**

He is pursuing his masters in software engineering from Delhi Technological University (formerly called Delhi College of Engineering), Delhi, India. He completed his graduation from YMCA Institute of Engineering, Faridabad, India. His research interests are software quality, software metrics and statistical and machine learning models.