Abstract
In Software engineering, there are a plenty of prediction approaches used for several purposes like fault prediction, security prediction, effort prediction, correction cost prediction, reusability prediction, test effort prediction and quality prediction. These approaches help to minimize the cost of testing which minimizes the cost of the project. In this paper, we study software fault prediction techniques to find the software defects at an early stage of software development life cycle. The methods, metrics and datasets are used to find the fault-proneness of software. The various techniques like linear regression, logistic regression, negative binomial regression, fuzzy subtractive clustering, radial basis function (RBF) network, multilayer perceptron, support vector machine, artificial neural networks, instance-based reasoning, Bayesian-belief networks, decision trees, rule induction, multi-linear regression models, multivariate models, back propagation neural Network (BPN), probabilistic neural network (PNN), expert estimation and nearest neighbor are used to predict the fault proneness of the software. Each of these has its own advantages and disadvantages.

Keywords
Statistical, Machine learning, Statistical and Machine learning, Statistical vs. Expert estimation, Nearest Neighbor

I. Introduction
To find the Software defects in the Software development life cycle before the testing process, Software metrics (method-level, class-level, component-level, file-level, process-level, quantitative values-level) are used. Methods used are statistics, machine learning, machine learning along with statistical methods and statistical models vs. expert estimation [9]. To predict the fault prone modules for the next release of software, industry uses previous software metrics and fault data [12]. Software fault prediction approaches are much more cost-effective to detect software faults compared to software reviews [12]. As the size and complexity of software systems increases, software industry is highly challenged to deliver high quality, reliable software on time and within budget. Prediction model for software defects helps to minimize the cost and improve the software quality.

II. Review of literature
A. Statistics Based Approach
To detect faulty software components during the earlier stage of software development life cycle and to build a prediction model using object oriented classes early to identify the faulty classes statistics based approaches are widely used. To identify the fault prone classes, export coupling metrics are extremely popular [2]. The prediction model has a high accuracy but has a low design quality. They collect the data from the commercial java applications and identify the fault classes for the future release of applications.
To compare the accuracy of fault prediction models available before and after the system is implemented design [2] and code metrics [1] are used. Code metrics are available only after the system is developed and design metrics are available before the coding has started [18]. Models are developed using linear regression and based on the data from one release of a large telecommunication system developed by Ericsson. In their study, prediction made after the system is 34% more accurate than before the system. The variability of metrics available before and after the implementation is 43% and 58% [18] but when metrics are not used the performance of the system is same.
The files contain the large number of faults in the next release of software system. To identify the number of faults in each file of the next release of a system, a negative binomial regression model is developed [3] that provides the number of faults for each file of a release based on characteristics such as
- The file size,
- The file was new to release,
- File was changed or unchanged from the previous release,
- The number of faults in previous release,
- The age of the file, and
- The programming language [3].
In their study, average percentage of faults identified by the model is 20% and the prediction of faults is accurate [3]. To identify the fault-prone software modules, whether software metrics are available early in the software development life cycle, a predictive model is built to increase the prediction accuracy using metrics that describes textual requirements, available static code metrics and combination of both requirement metrics and static code metrics. The researchers use three NASA projects data CM1, JM1 and PC1 [19]. Requirement metrics are useful for defect prediction.

B. Machine Learning Approach
To increase the software reliability and quality, a software fault prediction approach is the best approach to identify the faults. Data is divided into clusters using Fuzzy subtractive clustering and then the radial basis function (RBF) network is applied to predict software faults [13]. Accuracy of RBF is better than multilayer perceptron (MLP) but inspection cost and completeness value of MLP’s is higher [13]. To provide the software quality using median-adjusting class labels, a neural network classifier with metric inputs and class labels are used [14]. It uses
- Multilayer Perceptron
- Three multilayer analysis
- A test set of metrics with non-adjusted quality class labels [14].
Median-adjusted class labels are an effected preprocessing method to predict a software object quality. Performance of component-wise Median is poorly due to its lack of orthogonal invariance [14].
In object oriented software systems, to predict faults, a classification approach is used. Multilayer perceptron (MLP) is based on fault prediction model. Prediction of faults are analyzed in classification and further classified according to the type of fault [15]. It is based on clustering and radial basis function (RBF) network is used. In their study, MLP used to identify faulty classes and RBF Neural network categorize faults according to several fault types that are defined in faulty classes [15]. For early software quality prediction, when only a small number of sample data are available, a Support Vector Machine (SVM) based
software class model is developed. Several techniques are used to classify the software program modules into fault-prone and non-fault-prone categories [16]. SVM is a technique that is used for data classification. SVM is robust in nature than other techniques for software quality prediction. It is adaptive to modeling non linear functional relationships. It achieves a good performance and is an efficient technique for software quality prediction [16]. Researchers also apply a machine learning approach on real-time software systems to predict the software defects. Examples include tele-control/tele-presence, robotics and mission planning systems [17]. There are number of prediction techniques (Statistical models such as Stepwise Multi-linear Regression models and multivariate models, and machine learning models, such as Artificial Neural Networks, Instance-based Reasoning, Bayesian-Belief Networks, Decision Trees and Rule Induction) that are used but there is no such a technique that gives a accurate result for all the data sets. Size and complexity metrics are not sufficient for predicting real-time software defects.

Two neural networks, back Propagation Neural Network (BPN) and Probabilistic Neural Network (PNN) based software fault prediction models use object-oriented class metrics. They used the data generated in an academic institution. They compare the results of these two neural network models with statistical methods (discriminate analysis and logistic regression) using five quality model parameters [6]. This study shows that PNN is better than BPN and it is robust in nature.

To minimize the cost and improve the effectiveness of the Software testing process, they used Artificial Neural Networks (ANN’s) and Support Vector Machines (SVM’s) on a Data set obtained from NASA’s Metrics Data Program data repository [8]. There are so many Software metrics like Lines of Code (LOC), McCabe (1976), and Halstead (1977) metrics that are used in early research. They applied ANN’s and SVM’s to classify modules as faulty or error-free.

C. Statistical and Machine Learning Approach
To predict the fault proneness of the code in the Mozilla open source Software system, Researchers use Statistical methods (Logistic regression and Linear regression) and Machine learning techniques (Decision trees and Neural networks) [5]. According to them, performance of Lines of code (LOC) metric is well and correctness of Lack of Cohesion on Methods (LCOM) metric is good but its completeness value is low. Multivariate models perform better for fine grained analysis [5]. Researchers predict the Fault-prone Software modules using Statistical and Machine learning methods. They compare the logistic regression with Machine learning methods (Artificial Neural Network, Decision Tree, Support Vector Machine (SVM), cascade correlation network, group method of data handling polynomial method, gene expression programming) to find the impact on Static code metrics on fault proneness [11]. Researchers use Receiver operating characteristic (ROC) curves for the modules predicting as fault-prone or not fault-prone, to determine the performance by calculating Area under curve (AUC) from a ROC Curve [11].

D. Statistical Models Vs. Expert Estimation
Researchers compare the accuracy of Statistical prediction models with the expert estimation on two large telecommunication systems. To predict the Fault-prone of the code units they use two approaches, statistical fault prediction model and human experts [6]. Prediction on the class level and component level is also performed. Statistical models are cheap to build and they can be used in the absence of experts and they perform well on small and large systems. On large systems, expert estimation is limited [6].

E. Nearest Neighbor Techniques
There are so many sampling techniques that are used traditionally like random, stratified, systematic and clustered but all these techniques use the class attributes not an non class attributes [4]. So, they use 348 and Naïve Bayes on five NASA defect datasets. Each dataset is divided into three groups, a training set, nice neighbor test sets and nasty neighbor test sets. The result shows that accuracy of nice experiments is 94% and nasty experiments are 20% [4]. The K Nearest Neighbor (KNN) method technique is used in clustering and classification. Classification approaches based on KNN [20] are:
- The Density Based KNN Classifier (DB-KNN)
- The Variable K nearest Neighbor Classifier (V-KNN)
- The Weighted KNN Classifier (W-KNN)
- The Class Based KNN Classifier (CB-KNN)
- The Discriminability KNN Classifier (D-KNN)

In their study, due to the structural density calculation, DB-KNN method works slower than KNN but its performance is better than KNN. W-KNN classifier was fast and its performance was better than KNN. W-KNN method was exceptionally fast and its performance was better than KNN. CB-KNN is the most accurate classifier and D-KNN is a reliable classifier [20]. To reduce the memory requirement and computational complexity the nearest neighbor techniques are divided into two categories, like Structure less (Weighted KNN, Model based KNN, Condensed NN, Reduced NN, Generalized NN) and structure based (K-d tree, ball tree, Principal Axis Tree, Nearest feature line, Tunable NN, Orthogonal Search Tree) nearest neighbor techniques [10]. Structures less technique overcome the memory limitation and structure based technique reduce the computational complexity.

III. Conclusion
In this paper we see most of the techniques that are based on the machine learning approaches and used the NASA’s public datasets to predict the Software faults. Method level metrics are used instead of Class level metrics. Machine learning models have better features than statistical methods or expert opinion. So, it is found that machine learning models are mostly used and increase the usage of public datasets for fault prediction in future.

References


Malkit Singh received his B.Tech degree in Information Technology from Lovely Professional University, Phagwara, Punjab, INDIA, in 2011, now he is doing M.Tech (CSE) from Lovely Professional University, Phagwara, Punjab, INDIA. His research interests include software engineering.

Dalwinder Singh Salaria received his B.E. degree in CSE from SSJCOE, Jalgaon(MH), INDIA, in 2003 and the M.Tech degree in CSE from Guru Nanak Dev Engineering College, Ludhiana(PB), INDIA, in 2008. He is working as an Assistant Professor with Department of CSE at Lovely Professional University (PB), INDIA. His research interests include databases and Software engineering.