Volume 5, Issue 8, August 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Adaptive Boost Learning Approach: An Improved Method for Software Defect Prediction

Punika Mahajan
(M.Tech in Computer Engineering, University College of Engineering,
Punjabi University, Punjab, India

Abstract: ***An important goal during the cycle of software development is to find and fix existing defects as early as possible. Although, big software development companies have their own development repository, which typically includes a version control system and a bug tracking system. There is no doubt that these systems proved useful for software defects prediction. Still, lot of work has to be done for software defect prediction and management. In this report, two models has been proposed for defect prediction (i) Support Vector Machine (SVM) is combined with Principle Component Analysis (PCA) and (ii) Adaptive Boost of Support Vector Machine(SVM)- Radial Basis Function (RBF) is combined with Principle Component Analysis. Principle Component Analysis has been used for reducing data dimensionality. First, the defect prediction capability of different existing methods without and with Principle Component Analysis has been analyzed using WEKA and MATLAB. The comparative analysis of existing and proposed models demonstrate that the defect prediction capability of Adaptive Boost with SVM-RBF kernel approach is better.***

Keywords- Adaptive Boost, Data Mining, Feature Extraction, Machine Learning, Principle Component Analysis (PCA), Support Vector Machine (SVM), Radial Basis Function(RBF).

I. INTRODUCTION

It is valuable to predict the software that is defect- prone. There have been many studies and learning approaches that are used to measure the performance of software. Analysis of all required features of defect prediction are used to determine that what factors influence predictive performance. The quality of the software can be measured with the different features such as cyclomatic complexity, design complexity, effort, time estimator, length of the program, operands, operators, line count etc.

1.1 Software Engineering: Introduction

Software Engineering is defined as the systematic and well defined approach to the development, operation, maintenance and retirement of the software. By the word 'systematic' means that the methodologies used for the development of the software are repeatable. The goal of software engineering is to take software development closer to science and engineering that solves the problems of the clients and away from those approaches for development whose outcomes are not predictable. Software engineering approach is shown in fig 1.1

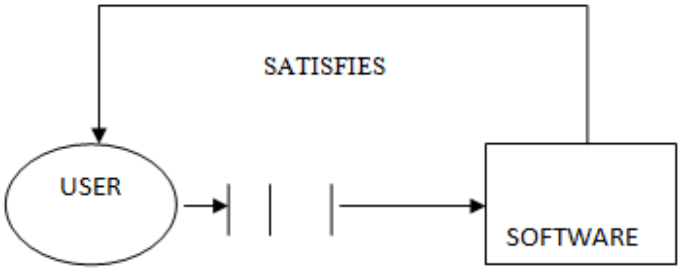


Fig 1.1 Software Engineering

1.1.1 Software Quality Attributes

Software Quality attributes can be defined as follows :

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

1. **Functionality** : It is the capability that provides functions which meets the defined and implied needs of the software when it is used.
2. **Reliability** : It is the capability that maintains the specified level of performance.
3. **Usability** :The capability to be understood , learned and used.
4. **Efficiency** : It is the capability to measure the performance relative to the amount of resources used.
5. **Maintainability** : It is the capability to be updated and modified for purposes of making corrections , improvements or adaptation.
6. **Portability** : It is the capability to be adapted for different environments without applying actions.

1.2 Defect Prediction in Software module

a) Data cleansing process

- Initial Preprocessing of the data
- Removal of Constant attributes
- Removal of Repeated attributes
- Replacement of Missing Values
- Enforce Integrity with Domain Specific Expertise
- Removal of Repeated and Inconsistent Instances

b) Prediction Performance measures

- Precision
- Recall
- Accuracy

c) Data Extraction

- Classifier family
- Data set family
- Metric family
- Researcher Group

1.3 Machine learning

Machine learning is a science that explores the building and study of algorithms that can learn from the data. Machine learning process is the union of statistics and artificial intelligence and is closely related to computational statistics. Machine learning takes decisions based on the qualities of the studied data using statistics and adding more advanced artificial intelligence heuristics and algorithms to achieve its goals. Machine learning tasks are classified into three broad categories :

- Supervised learning.
- Unsupervised learning
- Reinforcement learning

1.4 Data mining

Data mining is related with the discovery of new and interesting patterns from large data sets for analysis and executive decision making. Data mining is described as the union of past and current or recent developments in statistics, artificial intelligence and machine learning. Data mining process is shown in fig 1.2

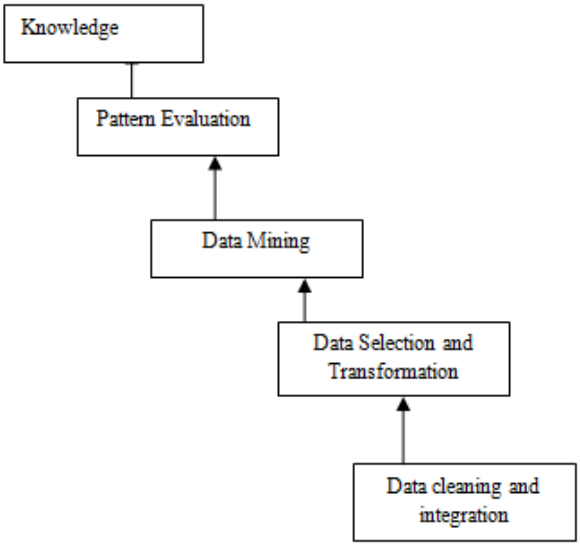


Fig 1.2 Steps for Extracting Knowledge from Data

1.4.1 Scope of Data Mining

- Automation in prediction of behavior and trends.
- Automated discovery of previously unknown patterns.

Automation in prediction of behavior and trends

This is the process of finding targeted information in large databases. Predictive problems include forecasting, insurance analysis for prediction and decision making, income tax department of government for fraud discovery.

Automated discovery of previously unknown patterns

This is the process of identifying previously hidden patterns in first step. Pattern discovery problems include detecting fraudulent credit card transactions and identifying anomalous data that could represent data entry keying errors.

1.4.2 Data Mining Process

Data mining consists of five major elements:

- Extraction and transformation of data onto the data warehouse system.
- Run data on multidimensional database system in a managed way
- Providing data access to business analysts and other professionals
- Data analyzing
- Presentation of data in useful and required formats such as tables and graphs.

1.4.3 Goals of Data Mining

- Prediction
- Identification
- Optimization
- Classification

Prediction : How certain attributes within the data will behave in the future.
Identification -- Use data pattern to identify the existence of an item, an event, or an activity
Optimization – Optimize the use of limited resources such as time, space, money, or materials and maximize output variables such as sales or profits under certain constraints.
Classification -- Partition data so that different classes or categories can be identified based on combinations of parameters.

1.4.4 Classification Algorithms

- Statistical Algorithms
- Neural Networks
- Genetic algorithm
- Decision trees
- Nearest neighbor method
- Rule induction

Statistical Algorithms :Systems like SAS and SPSS have been used by analysts to detect unusual hidden patterns and explain patterns using statistical models such as linear models.
Neural Networks : Artificial neural networks have the pattern-finding capacity. Neural Network algorithms can be applied to pattern-mapping. Neural networks works successfully on the applications that involves classification.
Genetic algorithms : Genetic combination, mutation, and natural selection in a design based on the concepts of natural evolution.
Decision trees : Tree-shaped structure represent sets of decisions . These decisions generate rules for the classification of a dataset. Specific decision tree methods include Classification and Regression Trees (CART) and Chi Square Automatic Interaction Detection (CHAID).
Nearest neighbor method : A technique that classifies each record in a dataset based on a combination of the classes of the k record(s) most similar to it in a historical dataset. Sometimes called the k-nearest neighbor technique.
Rule induction : The extraction of useful *if-then* rules from data based on statistical significance

1.5 Support Vector Machine (SVM)

Support vector machines (SVM's) are supervised learning models that analyze data and recognize patterns which are used for classification and regression analysis. Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on. Various types of SVM kernels are:

- Linear kernel
- Quadratic kernel
- Polynomial kernel
- Radial Basis Function kernel
- Multilayer Perceptron kernel

1.6 Adaptive Boost Learning Approach

Adaptive Boosting (AdaBoost) is a machine learning meta-algorithm that can be used in conjunction with many other types of learning algorithms to improve their efficient performance. The output of the weaker learning algorithms is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is sensitive to noisy data and outliers. Individually learners can be weak, but as long as the performance of each one is slightly better than random guessing, the final model can be proven to converge to a strong learner.

II. LITERATURE REVIEW

Many researchers have worked on the defect prediction of software module by using various methods and learning approaches . In this report, main focus is to reduce the complexity of processing by feature extraction method and boundary condition problem resolved as Support Vector Machine (SVM) and component learning by using adaptive boost with SVM -RBF(Radial Basis Function) Kernel. Their work is reviewed and defined as under:

David Gray et al. (2011) have explained the reason of significant preprocessing of data set for suitability of defect prediction. Researchers need to analyze the data that how it will be used by removal of constant attributes, repeated attributes, missing values and inconsistent instances. The experiments that have been used are based upon NASA metrics data program that results in errors findings and conclude that errors are mainly because of repeated data points[2].

Tim Menzies et al. (2011) describes that how to improve the effort estimates of a project and defect predictions of a software module. The best thing can do to control cost and defects is to discard the needless functionality by making the lines of code to minimum. It has been seen that local treatments are always superior and different to the global treatments because data that appears to be useful in global context is often irrelevant to the local contexts[3].

Qinbao Song et al. (2011) describes the framework that comprises scheme evaluation and defect prediction components. Analyzing the prediction performance for the given historical data sets is done by scheme evaluation and defect predictor constructs models according to the evaluated learning scheme and predicts defects of the software with new data according to the defined constructed model. It has been shown that different learning schemes should be used for different data sets[4].

Ming Li et al. (2012) states that software quality can be controlled by software defect prediction. The defect prediction techniques used currently are based on large amount of historical data but in case of new projects and for many organizations historical data is often not available. In that case , sample based methods for defect prediction can be used by selecting and testing a small percentage of module and after that build a defect prediction model to predict defect proneness of the other modules[7].

Martin Shepperd et al. (2014) have discussed about the factors having largest effect on the predictive performance of the software by conducting a meta analysis of all relevant and high quality primary studies of defect prediction on software module. The experimental results showed that the major factor is the researcher group instead of choice of classifier on the software performance[1].

G Czibula et al. (2014) focuses on the problem of importance during the time software evolution and maintenance for the problem of defect prediction. The quality of the software system is improved by identifying the defective software modules. Relational association rules are the extension of ordinal association rules that are being used in the prediction of software module that whether it is defective or not. It describes numerical range between the attributes that are commonly occur over a dataset by the experimental evaluation on the NASA datasets as well as a comparison is performed to similar existing approaches is provided[5].

A Okutan et al. (2014) states different software metrics that are used for defect prediction and defines the set of metrics that are most important for predicting the defectiveness in the software module. The two more metrics i.e. number of developers and the source code quality are defined other than the promise data set. Experiments results that lines of code and lack of coding quality are the most effective metrics whereas coupling between objects and lack of cohesion of methods are less effective metrics on defect proneness[6].

III. METHODOLOGY

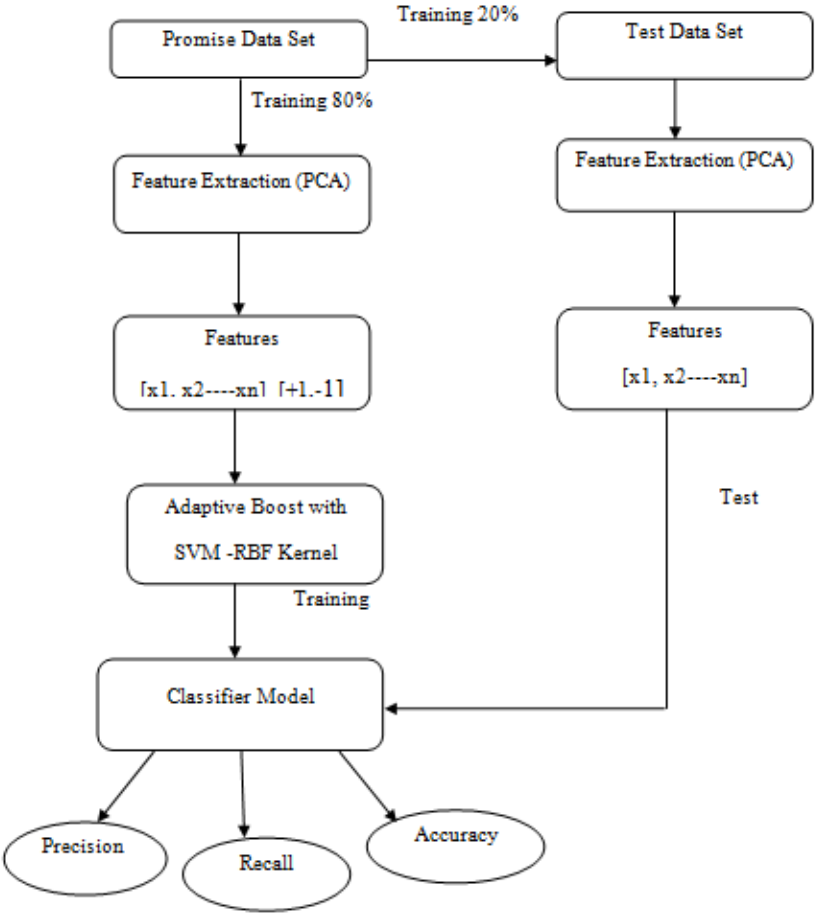
Step 1 : Take the promise data set with 21 different features like cyclomatic complexity , design complexity , effort , time estimator , line count etc for defect prediction in software module.

Step 2 : Implement feature extraction on promise data set by using Principle Extraction Analysis (PCA). Feature Extraction is used to merge the data set. In feature extraction merging process is based on eigen values , having high eigen value means contain more information.

Step 3 : Take the different features x1, x2, x3.....xn and find out the status that whether they are default or not default [+1, -1]. If the value is +1 that means its 'default' and if -1 then it is 'not default'.

Step 4 : Implement Hybrid Adaptive Boost with SVM -RBF Kernel for component learning and to remove compaction and boundary error condition.

Step 5 : Apply Classifier model to find out precision , recall and accuracy of the software module.



IV. RESULTS AND DISCUSSIONS

4.1Comparison Among Different Types of SVM'S Using Various Metrics on Cm1 Data Set of 21 Features (without PCA).

The defect prediction performance of different SVM's is analyzed using metrics like precision, recall and accuracy with the help of MATLAB.

Table 4.1 presents comparison among different SVM's on CM1 data set without feature extraction .Analysis among different types of SVM's in terms of various parameters on data set are shown in fig 4.1.

Table 4.1 Comparison among different SVM's with various performance metrics

SVM	**Precision**	**Recall**	**Accuracy**
Linear	69.5	61.99	67.98
Quadratic	80.93	69.02	74.72
Polynomial	85.72	74.2	82.58
RBF	84.7	72.89	80.9
Multilayer Perception	44.52	46.66	41.01
Adaptive Boost with SVM	**99.15**	**99.09**	**99.17**

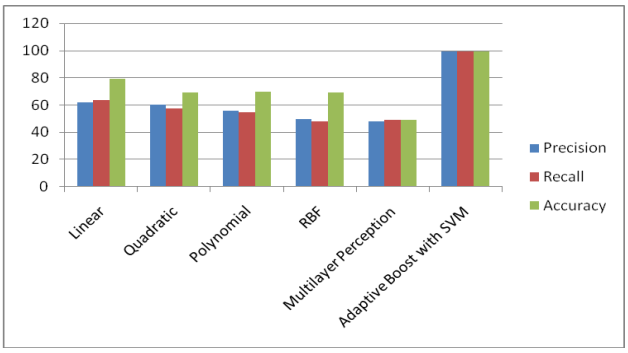


Fig 4.1 Comparison of various parameters of different SVM's

Above tables and graphs clearly depicts that Adaptive Boost with SVM learning approach is the best approach for the prediction of default and not default features of software model as precision, recall and accuracy is much better than all other learning approaches.

4.2 Comparison Among Different Types of SVM'S Using Various Metrics on Cm1 Data Set of 15 Features (with PCA)

Now the results are analyzed by reducing the data set from 21 features to 15 features. Analyzing of various parameters on data set will be done with the help of same SVM's that were used before.

Table 4.2 presents comparison among different SVM's on CM1 data set with feature extraction.

Analysis among different types of SVM's in terms of various parameters on data set (after feature extraction) are shown in fig 4.2 with the help of bar graphs.

Table 4.2 Comparison among different SVM's with various performance metrics

SVM	Precision	Recall	Accuracy
Linear	61.71	63.41	79.21
Quadratic	60.42	57.35	69.1
Polynomial	55.89	54.66	69.66
RBF	49.51	48.08	69.1
Multilayer Perception	48.1	48.87	48.88
Adaptive Boost with SVM	**99.15**	**99.09**	**99.17**

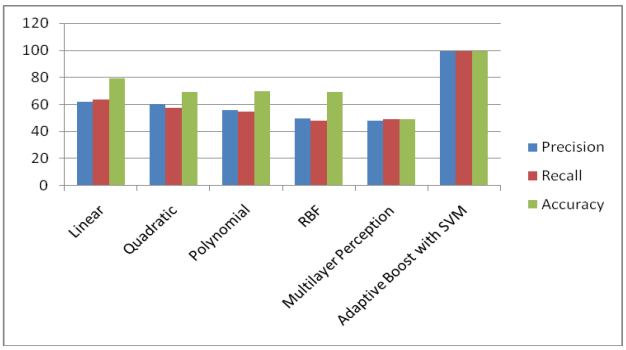


Fig 4.2 Comparison of various parameters of different SVM's (with PCA)

Even by reducing the features from 21 to 15 for analyzing the various parameters of software model, Adaptive boost with SVM (Adaboost) learning approach again gives better precision, recall and accuracy than the other approaches.

4.3 Comparison Among Different Types of SVM'S Using Various Metrics on Data Set (Cm1) Of 12 Features (with PCA) :

Now the results will be analyzed by reducing the data set from 21 features to 12 features. Analyzing of various parameters of data set will be done with the help of same SVM's that were used before.

Table 4.3 presents comparison among different SVM's on CM1 data set with feature extraction

Analysis among different SVM's in terms of various parameters on data set (after feature extraction) are shown in fig 4.3 with the help of bar graphs.

Table 4.3 Comparison among different SVM's with various performance metrics

SVM	Precision	Recall	Accuracy
Linear	58.93	61.28	78.65
Quadratic	57.6	56.55	72.47
Polynomial	57.06	56.59	73.6
RBF	49.51	48.08	79.21
Multilayer Perception	53.53	52.29	61.8
Adaptive Boost with SVM	**99.15**	**99.09**	**99.17**

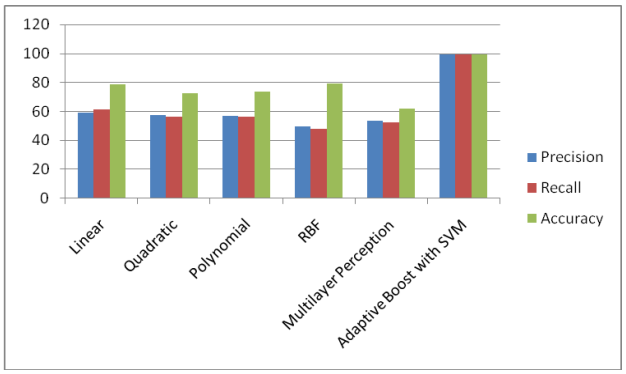


Fig 4.3 Comparison of various parameters of different SVM's (with PCA)

Even by reducing the features from 21 to 12 for analyzing the various parameters of data set, Adaptive boost with SVM (Adaboost) learning approach again gives better precision, recall and accuracy than the other approaches.

V. CONCLUSION

The construction of software cost estimation models remains an active topic of research. Knowing the estimated cost of a particular software project early in the development cycle is very important as management can use cost estimates to approve or reject a project proposal or to manage the development process more effectively. This research introduces a software performance prediction methodology using different machine learning algorithms. Many attribute selection techniques are applied on the datasets and the attributes are ranked with the Ranker Algorithm. The attributes of all the three datasets are chosen and machine learning techniques when applied to them (using WEKA) result in improvement of three important prediction performance measures i.e. precision, recall and accuracy. Implementation and evaluation of two models has been done for software defect prediction (i) Support Vector Machine (SVM) is combined with Principle Component Analysis (PCA) and (ii) Adaptive Boost of Support Vector Machine(SVM)- Radial Basis Function (RBF) is combined with Principle Component Analysis. Principle Component Analysis has been used for reducing data dimensionality. The defect prediction capability of different existing methods (without Principle Component Analysis) has been analyzed using WEKA and MATLAB. The comparative analysis of both the techniques, lead to the observation that the results obtained by Support Vector Machine- Radial Basis Function with Adaptive Boost learning approach is best among all other methods for prediction of default and not default features of software model as prediction performance measures i.e. precision, recall and accuracy of this approach is much better than all other learning approaches.

VI. FUTURE SCOPE

Based on the results presented in this work and the practical observations of software defect prediction, there are still some possibilities to extend the methods:

1. Extend the model to more software repositories.
2. Use more classifiers and more software metrics on data set to get more better data preprocessing method.
3. Considering the other possible factors for example: requirement analysis and software testing that can even build a more comprehensive and precise software defects prediction model.

REFERENCES

[1] Shepperd, Martin, David Bowes, and Tracy Hall. "Researcher bias: The use of machine learning in software defect prediction." *Software Engineering, IEEE Transactions on* 40.6 (2014): 603-616.

[2] Gray, David, et al. "The misuse of the nasa metrics data program data sets for automated software defect prediction." *Evaluation & Assessment in Software Engineering* (2011): 96-103.

[3] Menzies, Tim, Butcher, A., Marcus, A., Zimmermann, T., & Cok, D. "Local vs. global models for effort estimation and defect prediction." *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering* (2011): 343-351.

[4] Song, QinbaoJia, Z., Shepperd, M., Ying, S., & Liu, J. "A general software defect-proneness prediction framework." *Software Engineering, IEEE Transactions on* 37.3 (2011): 356-370.

[5] Czibula, Gabriela, Zsuzsanna Marian, and Istvan Gergely Czibula. "Software defect prediction using relational association rule mining." *Information Sciences* 264 (2014): 260-278.

[6] Okutan, Ahmet, and Olcay Taner Yıldız."Software defect prediction using Bayesian networks." *Empirical Software Engineering* 19.1 (2014): 154-181.

[7] Li, M., Zhang, H., Wu, R., & Zhou, Z. H. "Sample-based software defect prediction with active and semi-supervised learning." *Automated Software Engineering* 19.2 (2012): 201-230.

[8] Niuniu, Xie, and Liu Yuxun. "Review of decision trees." *Computer science and information technology* (2010) : 105-109.

[9] Goebel, Michael, and Le Gruenwald. "A survey of data mining and knowledge discovery software tools." *ACM SIGKDD Explorations Newsletter* 1, no. 1 (1999): 20-33.

[10] Hall, Tracy, et al. "A systematic literature review on fault prediction performance in software engineering." *Software Engineering, IEEE Transactions on* 38.6 (2012): 1276-1304.
[11] Gao, Kehan, et al. "Choosing software metrics for defect prediction: an investigation on feature selection techniques." *Software Practice and Experience* 41.5 (2011): 579-606.
[12] Liao, Shu-Hsien, Pei-Hui Chu, and Pei-Yuan Hsiao. "Data mining techniques and applications–A decade review from 2000 to 2011." *Expert Systems with Applications* 39.12 (2012): 11303-11311.
[13] Yang, Yongliang, S. James Adelstein, and Amin I. Kassis. "Target discovery from data mining approaches." *Drug discovery today* 17 (2012): S16-S23.
[14] Sharma, Kavita, Gulshan Shrivastava, and Vikas Kumar. "Web mining: Today and tomorrow." *Electronics Computer Technology, IEEE* (2011): 399-403.
[15] Shepperd, Martin, et al. "Data quality: Some comments on the nasa software defect datasets." *Software Engineering, IEEE Transactions on* 39.9 (2013): 1208-1215.
[16] Okutan, Ahmet, and Olcay Taner Yıldız. "Software defect prediction using Bayesian networks." *Empirical Software Engineering* 19.1 (2014): 154-181.
[17] http://rspublication.com/ijst/ijst%20pdf%20feb%2012/18.pdf
[18] Data mining , http://en.wikipedia.org/wiki/Data_mining
[19] Data mining Process, http://www.google.co.in/images.
[20] http://www.cs.umd.edu/~samir/498/10Algorithms-08.pdf
[21] http://www.cs.waikato.ac.nz/ml/weka/downloading.html
[22] https://en.wikipedia.org/wiki/Support_vector_machine
[23] https://www.google.co.in/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=principal+component+analysis+ppt
[24] https://en.wikipedia.org/wiki/AdaBoost
[25] http://www.unc.edu/~xluan/258/datamining.html
[26] http://page.mi.fu-berlin.de/rojas/neural/chapter/K7.pdf
[27] https://www.google.co.in/search?q=decision+trees
[28] https://en.wikipedia.org/wiki/Naive_Bayes_classifier
[29] http://www.solver.com/k-nearest-neighbors-classification-example
[30] http://promise.site.uottawa.ca/SERepository/datasets-page.html
[31] https://en.wikipedia.org/wiki/Machine_learning
[32] https://en.wikipedia.org/wiki/MATLAB

