International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 3, March - 2014

Software Defect Prediction System –Decision Tree Algorithm With Two Level Data Preprocessing

Reena P
Department of Computer Science and Engineering
Sree Chitra Thirunal College of Engineering
Thiruvananthapuram, India

Binu Rajan
Department of Computer Science and Engineering
Sree Chitra Thirunal College of Engineering
Thiruvananthapuram, India

***Abstract*— Software systems play an important role in our daily lives, and hence making quality software systems is a critical issue. A lot of work is being done in this area. One of the most important among them is Software Defect Prediction (SDP). Defect Prediction is a binary Classification problem where a particular software module will be classified as defective if the error is greater than 0 and as Non-Defective if the error is equal to 0. A number of software metrics and statistical models have been developed for this purpose. Majority of the Defect Prediction models are developed using Machine Learning techniques. The performance of different models can be compared using the parameters like Accuracy, Hit Rate and False Alarm Rate. This paper covers a literature review on the experiments done on the area of software defect prediction in the past few years and finally a new system is proposed.**

Keywords—Defect prediction; Attribute selection; Instance filtering; decision tree; Software metrics

I. INTRODUCTION

A software defect prediction framework refer to the system that can predict whether a given software module is defective or not. In general, a software defect prediction model is trained using software metrics and defect data that have been collected from previously developed software releases or similar projects. The model can then be applied to program modules with unknown defect data. The features or attributes of software defect prediction data sets influence the performance and effectiveness of the defect prediction model. Most of the experiments related with Defect Prediction are conducted in a Machine Learning tool or environment called WEKA and some are done in MATLAB. Since every organization will try to maintain their data secret, only few datasets are available for public which can be used for experiments. One of the most popular publicly available datasets includes MDP and PROMISE repository provided by NASA.

A. Metrics

Software metrics provide information for defect prediction. At present there are plentiful metrics for assessing software risks. And among them there are three categories that contain the most widely used metrics. They are McCabe, Halstead and lines of code (LOC) metrics. Metrics within these categories are listed below. Metrics that does not come under these three categories are given under others.

TABLE 1. SOFTWARE METRICS

Metrics Type	Definition
McCabe Complexity metrics	Cyclomatic complexity Design complexity Essential complexity
LOC	Loc_total LOC_blank LOC_code_and_comment LOC_comments LOC_executable Number_of_lines
Operator	Number_of_operands Number of _operators Number_unique operands Number_unique operators
Halstead metrics	Length Volume Level Difficulty Content Error Estimate Programming time

	Effort
Others	Branch count Call_pairs Condition_count Decision_count Decision_density/edge_count Global_data_complexity Global_data_sensity Maintenance_severity

B. Data Sets

The software data sets provided by the NASA IV&V Metrics Data Program – Metric Data Repository (MDP) are being used for majority of the experiments in software engineering related fields. The data repository contains software metrics as attributes in the data sets and also an indication of whether a particular set of data is defective or non-defective. All the data contained in the repository are collected and validated by the Metrics Data Program.

Some of the product metrics that are included in the data set are, Halstead Content, Halstead Difficulty, Halstead Effort, Halstead Error Estimate, Halstead Length, Halstead Level, Halstead Programming Time and Halstead Volume, Cyclomatic Complexity and Design Complexity, Lines of Total Code, LOC Blank, Branch Count, LOC Comments, Number of Operands, Number of Unique Operands and Number of Unique Operators, and lastly Defect Metrics; Error Count, Error Density, Number of Defects (with severity and priority information).

The details about some of the datasets used in the defect prediction techniques are given below:

TABLE II. DATA SETS OVERVIEW

Name	**Written In**	**#Modules**	**% of N-Def**	**% of Def**
CM1	C	498	90.16	9.83
JM1	C++	10885	80.65	19.35
KC1	C++	2109	84.54	15.45
PC1	C	1109	93.05	6.94

II. A REVIEW ON EXISTING DEFECT PREDICTION MODELS

A. A Software Defect Prediction System Using Two Level Data Preprocessing

A two level data preprocessing has been introduced in the field of software defect prediction [1].Two level data pre-processing is done along with four different K-NN classifiers and the same has been compared with the random forest classifier. The method used for pre-processing includes attribute selection and instance filtering. Attribute selection is carried out in order choose an optimal subset of attributes. Filtered Subset evaluation is used for attribute selection and Greedy stepwise algorithm is used for searching.

The use of different types of filtering before classification enhances the accuracy, AUC (Area under ROC curve) and PD (probability of detection) values. Accuracy is the ratio of correctly predicted and precision is the ratio of modules correctly predicted as defective to the number of entire module predicted as defective. The datasets used are CM1, JM1, KC1,PC1 and KC2 from NASA repository.

Four data analysis technique namely IB1, IBK, KSTAR and LWL have been carried out and are compared with rf(random forest). In instance filtering, Re- sample is used. The results are found better when both filters i.e. Attribute Selection and Instance Filter (Resampling) are used together, than using these filters separately. The performance measures considered for comparison are accuracy, AUC (Area Under Curve), and PD (Probability of detection) . Except LWL algorithm, all other algorithms discussed above achieved better performance with two level preprocessed data sets. A detailed performance comparison of all the above mentioned techniques with and without preprocessing has been given in the paper. The two filters were used independently and in combined form in experiments. The performance was better when the two filters were used independently. Accuracy was improved from 88.15% to 94.37% when single level preprocessing (Attribute Selection) was moved to two levels of preprocessing.

B. A General Software Defect-Proneness Prediction Framework

The defect prediction framework presented in this paper involves two stages, evaluation and prediction [2]. A learning scheme consists of: A data preprocessor, an attribute selector and a learning algorithm. Best learning scheme is selected in the evaluation stage and in the prediction stage the best learning scheme is used to build a predictor with all historical data. Here historical data is used to build the learner and entirely new data is given for testing. A wrapper method is used for preprocessing where preprocessing is not done independently over the dataset but is bind with the learning algorithm. MGF's study [6] was taken as baseline experiment and all the comparisons were performed with it.

12 learning schemes, two feature selectors, and three classification algorithms are used. For attribute selection, two different strategies based on greedy algorithms (Forward selection and Backward elimination) are used. Three learning algorithms used are Naive Bayes (NB), J48 and OneR. Twelve learning schemes resulting from the combination of two data preprocessors, two attribute selectors, and three

learning algorithms yields a total of 12 different learning schemes which are : *NB + Log + FS , J48 + Log + FS , OneR + Log + FS , NB + Log + BE , J48+ Log + BE , OneR + Log + BE , NB + None + FS , J48 + None + FS , OneR + None + FS , NB + None + BE , J48 + None + BE , OneR + None + BE.* For each data set, one set of values were taken as log of their actual values and same learning algorithms are applied over them. 17 datasets are used in the experiments out of which 13 are taken from NASA MDP repository and remaining 4 from PROMISE repository. The measures used for evaluating performance include AUC (Area Under ROC curve), Balance and Diff. Diff was introduced to compare the performance of this framework with the one proposed by MGF [6].

Analyzing the experiment results, the mean prediction performance of framework proposed in this paper is better than that of MGF [6] , indicating improved performance in both evaluation and prediction. This paper concludes that no learning scheme dominates, and different learning schemes for different data sets should be chosen. And consequently the evaluation and decision process is important.

C. *Effective Estimation Of Module's Metrics In Software Defect Prediction*

The existing and already proposed methods for defect prediction use datasets available in NASA MDP repository. These datasets includes a combination of high level design and code metrics for different modules in the specified project. Finding out the pattern in the values of these metrics and dependency among them is not an easy task. For some matrices it's easy but very complex for others.

The defect prediction technique introduced in this paper allows the user to estimate the value of hard -to-obtain features (Type –1– Features) from its specified determinants (easy to obtain features or Type–2–Features) for any of modules [5]. Some probable hidden relations among different metrics are discovered and an estimation system is built to estimate some metrics' values from a combination of others. So the user is required to provide not many metric values.

The defect prediction system consists of three key components: Approximate Dependency Miner, The Estimator Part and Fuzzy Rule-Based Classifier. AD-Miner (Approximate Dependencies) is used in the dependency mining part of the system. A functional dependency (FD) is said to be valid in a given relation r over R if :

$t[X_1]=u[X_1]$ for all Xi in X which implies t[A]=u[A] where t[X] is the value assigned to the attribute x of tuple t.

Any existing dependency between the values of Type-1-Features and Type-2-Features are discovered in this part using AD miner algorithm.

In the fuzzy estimation part, Wang and Mendel's fuzzy rule learning method is used to develop a set of fuzzy modeling systems with similar structures. These fuzzy modeling systems are used to estimate the value of Type-2-Feature using a combination of Type-1-Feature as its determinants once the user provide the Type 1 features as input to the classifier. The input is provided online.

The MSE(Mean Square Error) value for each features are determined and are used for result evaluation. NASA data sets are used for the experiment. It's observed that using this system all the hard-to-measure features are automatically estimated with high accuracy. Evaluation results of estimation system is shown in the below table.

TABLE III. RESULTS OF EFFECTIVE ESTIMATION [5]

Estimation(Type-1-Feature --->Type-2-Feature)	**MSE on Train data**	**MSE on Test data**
Branch_Count → Cylomatic_Complexity	0.05	0.09
Branch_Count,Num_Operands→ Design_Complexity	0.07	0.1
Branch_Count,Loc_Blank→ Essential_Complexity	0.04	0.04
Num_Unique_Operands,Loc_Executable→ Halstead_content	0.81	0.93
Branch_Count , Loc_Blank → Halstead_difficulty	0.35	0.38
Branch_Count , Loc_Total → Halstead_Effort	4.82	6.03
Num_Operands , Loc_Executable → Halstead_Error-Est	0.009	0.01
Num_Operands,Num_Operators→ Halstead_Length	1.24	1.11
Branch_Count , Loc_Blank → Halstead_Level	0.02	0.08
Num_Unique_Operands,Branch_Count→ Halstead_ProgTime	3.5	5.13
Num_Operands,Num_Operators→ Halstead_Volume	2.8	3.43

D. *Software Defect Prediction: Heuristics For Weighted Naïve Bayes*

A Naïve Bayes classifier assumes that the presence or absence of a particular feature is not related to the presence or absence of any other feature, when the class variable is given. It stresses upon the equal importance and independence of every features [8]. But this may not work well in all cases. And an extension of Naïve Bayes predictor which is built on weighted features called Weighted Naïve Bayes is introduced in this paper. The main aim is to use the software metrics depending on their importance in defect prediction. Weight Assignment is done in three ways : GainRatio Based WA, InfoGain Based WA and PCA Based WA. GainRatio and InfoGain are mainly used in decision tree construction. Both the above ranking estimates are converted into feature weights. Experiments were performed on 8 data sets available in NASA MDP repository and were carried out in MATLAB.

Performance measures considered are probability of detection (pd), probability of false alarm (pf) and Balance. An ideal case is to maximize pd and minimize pf. The performance measure balance is used to choose the optimal

(pd, pf) pair. The performance of Standard Naïve Bayes was compared with the three weighted Naïve Bayes. The performance of Standard Naïve Bayes and PCA based WA were outperformed by other methods. GainRatio Based WA and InfoGain Based WA approach showed better performance. That is non-linear methods for feature weighting give very good performance improvement over Naïve Bayes compared to linear methods.

E. Emperical Assessment Of Machine Learning Based Software Defect Prediction Techniques

A detailed analysis of some of the existing machine learning techniques for Defect Prediction has been carried out [4]. Four different data sets namely CM1, JM1 , KC1 and PC1 from NASA MDP repository are used for the same. 70% of the data was used as training data and 30% as test data.

Experiments were conducted in WEKA machine learning tool kit and SAS tool. The following learning methods were considered for experiment :

1. Decision Trees – J48 Trees
2. Naïve Bayes Classifier
3. Logistic Regression – LoR ,
4. Support Vector Logistic Regression - SVLR
5. Neural Network for Discrete goal field - NND
6. I-Rule (IR)
7. Instance Based Learning for 10 nearest neighbors (IBL)

MAE (Mean Absolute Error) was considered for performance assessment. Based on analysis , NB , IBL and NND performed better than other prediction models discussed above. Moreover the paper conclude that the selection of best learning technique depends on the data in sight at that point in time.

F. How Many Software Metrics Should Be Selected For Defect Prediction

Software metrics are collected during various stages of development for various purposes. Performance of a Software Defect Prediction Model can be improved significantly if we choose the software metrics wisely. The proposed technique centers on threshold based feature selection technique to remove irrelevant and redundant features [7]. The process of removing the redundant and irrelevant features and choosing the highly informative features is called feature selection. A feature selection technique can either be associated with a learner or can be applied independently.

In this proposed feature selection technique, first of all each attributes values are normalized between 0 and 1. And these values are considered as posterior probabilities. Each non-class attribute is then paired individually to the class attribute. A Threshold Based Feature Selection technique (TBFS) is conducted. Performance Metrics considered are Mutual Information (MI), Kolomogorov-Smirnov (KS) , Deviance (DV) , Area Under ROC curve and Area Under Precision Recall Curve(PRC). Mutual Information measures the mutual dependence of two random variables. Kolomogorov-Smirnov measure is used to measure maximum difference between the curves generated by the true positive and false positive rates as the threshold changes between 0 and 1. Deviance represents sum of squared errors from mean class.

The three classifiers used in the proposed framework are Multilayer Perceptron, k-Nearest Neighbors and Logistic Regression. Data sets for experiment were taken from eclipse project available in PROMISE data repository. All the experiments were conducted in WEKA tool. Classification models built with smaller feature subsets showed better performance than models with complete feature set.

G. Data Mining Static Code Attributes To Learn Defect Predictors

Defect predictors can be built with all available attributes followed by sub setting to find the most appropriate particular subset for a particular domain [6].

8 Data sets from NASA MDP repository were chosen for experiments in WEKA toolkit. The datasets consisted of entry for each module which describe the attributes for that module and number of defects in that module. During preprocessing, depending on the number of defects, that column was converted to Boolean attribute which describe whether its defective or not. Also all the data was passed through one of two filters: None (no change) or logarithmic filtering (Log of the number were taken). No other preprocessing is performed.

Three learning algorithms OneR, J48 and Naïve Bayes were applied on the preprocessed dataset. OneR stands for One Rule and it builds prediction rule based on simple threshold values of a single attribute, any one of software metrics in this case. In simple words OneR would build a defect predictor in the form of a decision tree of maximum length 1. J48 learner is JAVA implementation of decision tree algorithm. The algorithm recursively splits a data set according to tests on attribute values in order to separate the possible predictions and builds predictor in the form of a decision tree of any depth. The third classifier, Naïve Bayes is based on Bayes' Theorem. For building the defect predictor the posterior probability of the class attribute will be calculated and each module would be assigned to the class with highest probability.

The posterior probability is calculated using the formula:

$$P(H|E) = \frac{P(H)}{P(E)} \prod_i P(Ei|H) \quad \text{------> (1)}$$

Where Ei refers to fragments of evidence and P(H) refer to prior probability for a class.

The performance of the learners was assessed using ROC (Receiver Operator) curves, Probability of Detection (Pd) and Probability of False alarm (Pf) and Balance. Probability of Detection (Pd) and Probability of False alarm (Pf) are respectively the Y–axis and X- axis of ROC curve. A good predictor should have *Pd* very high and *Pf* very low. Two performance factors were considered for final performance assessment of learners and are : Balance(bal) and *Pd*. Balance or *bal* is calculate using the formula :

$$Balance = 1 - \sqrt{(0 - pf)^2 + (1 - pd)^2} \Big/ \sqrt{2}$$

------> (2)

The table given below gives a performance comparison of all the three learning algorithms.

TABLE IV.PERFORMANCE OF DIFFERENT ALGORITHMS [6]

Preprocessing	Method	Median
Pd		
LogNums	Naïve Bayes	52.4
None	Naïve Bayes	0.0
None	J48	0.0
LogNums	J48	0.0
None	OneR	-16.7
LogNums	OneR	-16.7
NotPf=100 –Pf		
LogNums	J48	0.0
None	J48	0.0
LogNums	OneR	.3
None	OneR	.3
None	Naïve Bayes	-2.3
LogNums	Naïve Bayes	-26.0
Balance		
LogNums	Naïve Bayes	22.1
None	Naïve Bayes	3.7
None	J48	0.0
LogNums	J48	0.0
LogNums	OneR	-11.8
None	OneR	-11.8

All the 38 attributes were used in the initial stage to build the defect predictor using three learning algorithms. After that subsetting was done in order to find out the most suitable particular subset for a particular domain. Naïve Bayes with log-transform performed best when all the 38 attributes in the dataset were considered. J48 Decision Tree algorithm performed better than OneR learner. Out of the six methods described above only Naïve Bayes with LogNums had a median performance both large and positive. Finally MGF concludes that a defect predictor cannot be assessed using a single data set and only one learner.

III. CONCLUSION AND FUTURE WORK

Defect Prediction is a two class classification problem which would classify a software module as Defective (If error count > 0) and Non-Defective (If error count = 0). The two pre requisites are source code and software metrics (Static code Attributes) for the module.

A defect prediction system based on decision tree algorithm can be implemented as a future work which is expected to perform better than all the techniques described above. The system will consists of two components: A preprocessor and a Classifier.

Four data sets from NASA MDP repository are to be collected for the experiment and they are CM1, PC1, JM1 and KC1. The data set contain continuous valued attributes. The preprocessor is responsible for preprocessing of data before loading them into the classifier. The redundant and missing data are already preprocessed in the MDP repository. The preprocessor in the proposed system consist of two steps : Attribute Selection and Re-Sampling. Attribute selection is carried out in order to reduce the size of attribute space by removing irrelevant and less informative attributes. The attribute selection in the proposed system is done based on correlation among attributes. Pearson's correlation coefficient is chosen for the same as it works well on continuous valued attribute values. Pearson's coefficient can be calculated using the formula:

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

------> (3)

Re-Sampling is done in order to reduce the class imbalance problem. Out of two approaches oversampling and under sampling, under sampling the minority data has been chosen for Re-sampling in the proposed framework.

Once the data is preprocessed, it is to be fed into the decision tree learner. Here 70% of the historical data used for training and the rest 30% for testing or feeding into the classifier. For building the decision tree, first step is to rank the attributes based on their priority or frequency of their values. The rank of attributes can be calculated using their entropy InfoGain (Information Gain) values. Information Gain of an attribute can be calculated using the equation:

$$Gain(S, A) = Entropy(S) - \sum_{v \in value(A)} \frac{|Sv|}{|S|} Entropy(Sv)$$

------>(4)

Decision tree is a classifier in the form of a tree structure. Decision tree classify instances by starting at the root of the tree and moving through it until a leaf node. The attribute having the highest information gain would be placed at the root of the decision tree. The construction of tree can be stopped when all the selected attributes has already been included along the path.

The performance measures to be considered are Accuracy, Pd (Hit Rate), Pf (False Alarm rate) and RMSE (Root Mean Square Error). For an ideal case Pd should be high and Pf should be low. These measurements are given by:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

------> (5)

$$Precision = \frac{TP}{TP + FP}$$

------> (6)

Where TP refers to True Positive and

TN refers to True Negative.

Once the proposed system is built on standard data sets, its performance can be compared with already established techniques for defect prediction.

REFERENCES

[1] Rashmi Verma, Anil Gupta, "Software defect prediction using two level data pre-processing"

[2] Qinbao Song, Zihan Jia, Martin Shepperd, Shi Ying, and Jin Liu "A General Software Defect-Proneness Prediction Framework" , IEEE Transactions On Software Engineering, Vol. 37, No. 3, May/June 2011.

[3] Kehan Gao, Taghi M. Khoshgoftaar, Huanjing Wang and Naeem Seliya "Choosing software metrics for defect prediction: an investigation on feature selection techniques"

[4] Venkata U.B. Challagulla, Farokh B. Bastani, I-Ling Yen "Empirical Assessment of Machine Learning based Software Defect Prediction Techniques" , 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, 2005

[5] S.M. Fakhrahmad, A.Sami , "Effective Estimation of Modules' Metrics in Software Defect Prediction" , WCE 2009, July 1 - 3, 2009

[6] Tim Menzies, Jeremy Greenwald, and Art Frank "Data Mining Static Code Attributes to Learn Defect Predictors" , IEEE Transactions On Software Engineering, Vol. 33, No. 1, January 2007.

[7] How many software metrics should be selected for defect prediction - huanjing wang ,taghi m. khoshgoftaar and naeem seliya. Proceedings Of The Twenty Fourth International Florida Artificial Intelligence Research Society Conference.

[8] Software Defect Prediction: Heuristics for weighted Naïve Bayes – burak turhan , ayse bener. PROCEEDINGS OF WORLD CONGRESS ON ENGINEERING 2009 VOL 1