Journal of Theoretical and Applied Information Technology
31st December 2017. Vol.95. No 24
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IMPLICATIONS OF DISCRETIZATION TOWARDS IMPROVING CLASSIFICATION ACCURACY FOR SOFTWARE DEFECT DATA

[1]POOJA KAPOOR, [2]DEEPAK ARORA, [3]ASHWANI KUMAR

[1,2]Department of Computer Science & Engineering, Amity School of Engineering & Technology, Amity University, India

[3]IT and Systems, Indian Institute of Management, Lucknow, India

E-mail: [1]inkhanna@gmail.com, [2]deepakarorainbox@gmail.com, [3]ashwani@iiml.ac.in

ABSTRACT

Since the advent of new software architectures, paradigms and technologies the software design and development has developed a cutting edge requirements of being on the right track in terms of software quality and reliability. This leads the prediction of defects in software at its early stages of its development. Implications of machine learning algorithms are now playing a very crucial role in classification and prediction of the possible bugs during the systems design phase. In this research work a discretization method is proposed based on the Object Oriented metrics threshold values in order to gain better classification accuracy on a given data set. For the experimentation purpose, Jedit, Lucene, tomcat, velocity, xalan and xerces software systems from NASA repositories have been considered and classification accuracies have been compared with the existing approaches with the help of open source WEKA tool. For this study, the Object Oriented CK metrics suite has been considered due to its wide applicability in software industry for software quality prediction. After experimentation it is found that Naive Bayes and Voted Perceptron, classifiers are performing well and provide highest accuracy level with the discretized dataset values. The performance of these classifiers are checked and analyzed on different performance measures like ROC, RMSE, Precision, Recall values in this research work. Result shows significant performance improvements towards classification accuracy if used with discrete features of the individual software systems.

Keywords: *Discretization, Software Defect Prediction, Classification, CK metrics*

1. INTRODUCTION

Software defect prediction is the process of identifying probable defective modules in the software, prior to the testing phase. Software defect prediction plays an important and critical role, especially in cases where the software development time is a crucial issue and or where developed software system is too large for performing exhaustive testing [1-2]. Various Software metrics like the size of code, complexity of code, coupling, cohesion, Number of children, Depth of Inheritance, previous releases of software with the logs of defects encountered are being used to estimate the quality of the final software system. Efficiency of the available metrics in software defect prediction and estimation of the final software quality has been analyzed by various studies in the literature [1].

Different works in the literature have tried to address various issues related to software defect prediction but no standard measures for performance assessment are available for software quality prediction and there exists a lack of generic framework of defect prediction at the early stages of software development. Most of the studies in literature have used different data sets and suggested their own methodology for solving various problems related to software defect prediction [3-8]. Many of the studies use the CK defined metrics suite, considering the already established co-relation of CK metrics with software quality attributes like maintainability, re-usability, testability, complexity etc. CK suggested that value of these six metrics in suites: wmc, noc, dit, rfc, lcom, cbo should be maintained to low, medium, high range using software designs so that it meets the software quality requirement specifications. But the study does not define values for the linguistic terms used here: low, medium, high. Classes with large number of methods are considered to be more application specific but restricting the Re-use of code, so low wmc (Weighted Methods per Class) is desirable if Re-Usability is the desired quality attribute of the quality requirement specifications [9].

A threshold based discretization method applicable on CK metric has been suggested and described in this study. The CK metric data has been collected from promise repository for six metrics of CK suite. The data is in continuous form, and the study suggests adoption of a transformation method for discretizing this data will ensure a better defect prediction. Two classifiers have been used for prediction. These classifiers are the Inductive learning systems that can be used for classification of input data to a particular output class. The output class in the study has True / False value for the status of occurrences of software fault. The two classifiers used for the work are Naive Bayes and Voted Perceptron from the WEKA 3.6 Tool. Various statistical measures like ROC, TP, Accuracy, Kappa, F-measure, relative absolute error have been used to analyze the suggested discretization method. Threshold based discretization show a significant improvement in the classification performance of the two existing learning algorithms, the Naive Bayes and the Voted Perceptron. Though for SOM classifier, authors didn’t find any remarkable difference. The rest of this paper is organized as follows: In section 2, Background studies are discussed. In section 3, the proposed discretization method for data transformation is presented. In section 4, data set description is discussed for the analysis of the proposed method. In section 5, experimental design is discussed .Results of the proposed methodology are discussed and compared using the various statistical measures in section 6. Section 7 finally presents the conclusions of the study.

2. BACKGROUND

A large amount of the total development cost is kept aside for the testing phase. But sometimes due to size complexity exhaustive testing becomes a hard problem with time and cost constraints Software Defect Prediction (SDP) is an important activity before the testing phase that identifies probable defective modules. But defect prediction is not always an easy job to do [10]. Software Defect prediction (SDP) faces various issues to come up with an accurate Defect Prediction model [1]. The study has done an exhaustive research on how to address those issues like: Relationship between Attributes and Fault, No Standard Measures for Performance Assessment, Issues with Cross-Project Defect Prediction, No General Framework Available, Economics of Software Defect Prediction and the Class Imbalance Problem [11and 13]. Various studies on prediction models have been reported in the literature [12, 14 and 16]. An important study [14] presented a model that can predict the defective module on the basis of complexity metric like size complexity. The study proposed a direct relation between the complexity of code and defects. Though the paper was unable to present an accurate model, it raised many research questions. The paper also concluded that better metric for defect prediction model need to be identified and a standard process should be evolved, that can be integrated with the software development process to ensure a better software defect prediction at early stages of software development [14]. Another study [6] presents a data mining approach that can predict the defective state of software modules. The study shows better prediction capabilities when all the algorithms are combined using weighted votes. In the paper, the authors have introduced kernel-based asymmetric learning for software defect prediction. To eliminate the negative effect of class imbalance problem, the author proposed two algorithms called the asymmetric kernel partial least squares classifier and the asymmetric kernel principal component analysis classifier [6]. The use of the CK software metric suite in predicting the maintainability of object oriented software metrics threshold has been presented in another study [19]. The paper identified that the maintainability can be predicted on the basis of the most effective metric out of the dozens available metrics [19]. In survey it is found that researchers have explored all possible area to improve classification accuracy for software fault prediction. In study supervised discretization technique developed by Fayyad and Irani was used to see the effect of discretization process on classification accuracy using Naïve Bayes and J48 classifiers. The study concluded significant improvement in classification accuracy of the two classifiers used the study [25].

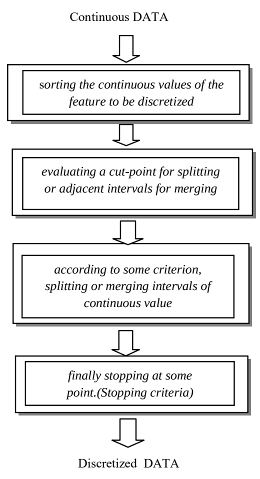
Continuous DATA
sorting the continuous values of the feature to be discretized
evaluating a cut-point for splitting or adjacent intervals for merging
according to some criterion, splitting or merging intervals of continuous value
finally stopping at some point.(Stopping criteria)
Discretized DATA

Figure1: Steps involved in Discretization process

3. PROPOSED DISCRETIZATION METHOD TRANSFORMATION

Discretization is the process of transferring continuous functions, models, and equations into equivalent discrete values. Discretization of continuous features or attributes, play an important role in the Machine learning data pre-processing phase. Many machine learning algorithms even perform their own discretization technique for classification. In WEKA, various classification techniques perform the Discretization transformation before actually performing classification. However, deciding a CUT-Point for discretization is most crucial issue for good classification accuracy and achieving a better efficiency of any Machine learning algorithm. CUT-POINT can be defined as a real value from the range of continuous values such that it divides the range into two intervals. Various studies have achieved better classification accuracy on the basis of discretized data [17]. A Discretization process generally includes four basic steps as depicted in Figure 1:

The first step requires the data to be in order, than a CUT POINT is selected to split the ordered data. A control strategy is decided for actually splitting of the data. Equal width binning and equal frequency binning, Error based discretization, Recursive minimal entropy partitioning are some of the discretization methods used by various studies in the literature for continuous feature discretization [21, 23]. A special case of discretization "Dichotomization" in which the number of discrete classes is 2, which can approximate a continuous variable as a binary variable. Dichotomization divides the entire range of the feature in 2 classes, making the data set more suitable for binary classification [18, 21].

$$f(x_{ij}) = \begin{cases} 1, & T1 < x_{ij} < T2 \\ 0, & else \end{cases} \text{---------------(1)}$$

where *i={1.....n}, n total number of rows in data set, j={1.....m}, m attributes in the data set.*

In equation (1), x_ij, x represents value of ith row data and jth attribute. {T1, T2} are defined as MMV± Deviation of jth attribute of the data set. Consider a case from Jedit3.2 data set, x2wmc=14 where j is wmc and i=2. MMV for wmc as given by the study is 13, if deviation for this wmc attribute comes out to be 3, then f(x2wmc) = 1, as T1= (13-3=10) and T2 = (13+3=16). Using this method, the entire data set is transformed and the efficiency of two different classifiers has been checked and the results compared using the different statistical measures.

Table 1: Mean Metrics value identified by the study using the Naive Bayes Classifier [15].

Metric	Mean Metrics Values (MMV)
WMC(mean)	11
DIT	2
NOC	.5
CBO	13
RFC	40
LCOM	-
Instances	900

4. DATA SET DESCRIPTION

The software defect data set for the study has been collected from the NASA repository (promise repository)[24]. The data set contains a total of 21 attributes, but for the current study 5 metrics of the CK metric suite i.e. wmc, dit, noc, cbo, rfc along with the output class attribute fault have been considered. Majority of the researchers have used CK metric for quality prediction for design purpose. CK metric suites are widely accepted for checking the quality of the final product. CK suite is a collection of six metrics, however only five metrics of CK metrics suite have been considered as suggested [15]. Five metrics chosen for the study from the CK metric suite are of Numeric type. Table 2 lists all the eighteen defect data sets used in the study. Attributes considered in the table are No of instances in each data set (#) and total distinct values for each metric in the eighteen different data sest. For example Jedit 3.2 has 287 as the total no of instances (#) and distinct values for each column, while WMC=44, DIT=11, NOC=13, CBO=46, RFC=93.

Table 2: Description of the eighteen defect data sets used for the study

		WMC	DIT	NOC	CBO	RFC
DEFECT SET	#	distinct values	distinct values	distinct values	distinct values	distinct values
Jedit 3.2	287	44	11	13	46	93
Jedit 4.0	306	43	8	12	45	97
Jedit 4.1	312	46	8	12	46	92
Jedit 4.2	369	53	8	12	55	114
Jedit 4.3	492	54	8	12	60	127
Lucene 2.0	195	30	5	10	34	57
Lucene 2.2	247	33	5	11	40	65
Lucene 2.4	340	39	5	12	43	73
tomcat	858	76	6	16	52	136
Velocity 1.4	196	34	4	7	39	65
Velocity 1.6	229	35	5	9	39	65
Xalan 2.4	723	69	8	20	76	122
Xalan 2.5	803	73	8	21	74	123
Xalan 2.6	885	76	8	21	72	125
Xalan 2.7	911	75	9	21	79	133
Xerces 1.2	440	52	6	11	35	82
Xerces 1.3	453	56	5	11	35	81
Xerces 1.4	588	50	5	12	45	87

5. EXPERIMENTAL DESIGN

The primary objective of the study is to discretize the CK metric using the proposed discretization method. For the purpose of study the we are concentrating on CK metric namely cbo, rfc, wmc, dit and noc. In the study, a co-relation between these five metrics and occurrence of fault has been established. lcom is not considered because of its unpredictable response to fault. The study also suggested a Mean Metrics Value (MMV) on the basis of the Naive Bayes classifier and study established a relation that if over all distribution of CK metrics suite for a system is as per MMV, it ensures less occurrence of faults [15]. This MMV has been used as CUT-POINT for discretizing the data. The value of f(x_ij) is set 1 if x_ij lies between T1 and T2. Value of T1 and T2 is (MMV± deviation).

Figure 2 represents the output after the discretization f(x_ij) is applied on a set of ten instances {I1,I2,I3,I4,I5,I6,I7,I8,I9,I10} and five attributes{X1=wmc,X2=dit,X3=noc,X4=cbo,X5=rfc}. The deviation corresponds to each attribute column calculated considering all the 340 instances. WEKA 3.6 has been used for the empirical study, using the Naive Bayes and Voted perceptron classifiers. The primary objective of the study is to discretize the continuous features of the data and then analyze the performance of the Naive Bayes and voted perceptron classifier.

6. RESULTS AND DISCUSSION

Eighteen different defect data sets, collected from the NASA repositories of six different systems and their versions, have been used to test the effect of the proposed process of discretization of continuous features of the data set. Looking at the continuous data set, it is visible that in some cases the number of distinct values for a feature is too high, and for classifiers like Naive Bayes and neural network it increases the computational complexity [23]. Jedit 3.2 has 196 as total number of instances and RFC feature contain 97 different values i.e 32% different values of total instances. Similarly in case of velocity 1.4, RFC feature contains 33% distinct values.

The classifiers were run for eighteen different defect sets and different statistical measures are recorded, in Table 3 for Naive Bayes and Table 4 for Voted Perceptron. The statistical measures: ROC, TP, Accuracy, Kappa, F-measure, relative absolute error, recall have been considered for comparing the classifier performance [20]. Accuracy is a measure of correctness of the classifier i.e. a classifier is able to predict an instance to TRUE class given it is TRUE and an instance predicted to False class given it is actually FALSE. Figure 3, Figure 4 and Figure 5 represents the accuracy performance of Naive Bayes, Voted perceptron and SOM.

Figure 3 represents the accuracy comparison of discrete and continuous data obtained using th e Naive Bayes classifier. Graph presents better accuracy in case of discretized data for both the classifiers. Though from Figure 3, it is quite obvious that in some cases like for lucene 2.0, the Naive Bayes classifier accuracy for continuous data is better than the accuracy obtained through discredited data set. Similarly, results obtained in Figure 4 for Voted Perceptron classifier indicates an overall accuracy improvement, except few cases like in xerces 1.4, where the graph shows better results for continuous data set in this case.

wmc	dit	noc	cbo	rfc	fault
5	2	0	19	18	y
10	3	0	13	41	y
10	1	0	4	38	n
4	4	0	3	7	n
25	2	2	20	82	y
5	1	3	13	6	n
13	1	0	13	61	y
17	2	0	6	34	y
13	1	0	8	36	n
8	1	0	30	8	n

$\Rightarrow$

Wmc	Dit	noc	cbo	rfc	Fault
1	1	1	1	0	Y
1	0	1	1	1	Y
1	0	1	0	1	N
1	0	1	0	0	N
0	1	0	1	0	Y
1	0	0	1	0	N
1	0	1	1	0	Y
1	1	1	1	1	Y
1	0	1	1	1	N
1	0	1	0	0	N

Figure 2: To show transformation of continuous data into discrete data by using the proposed discretization function $f(x_{ij})$

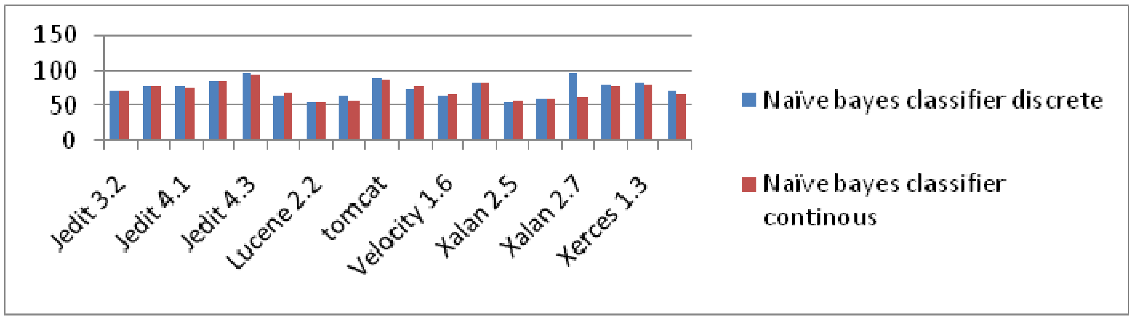

Fig 3: Accuracy comparison of Discrete and continuous data using the Naive Bayes classifier

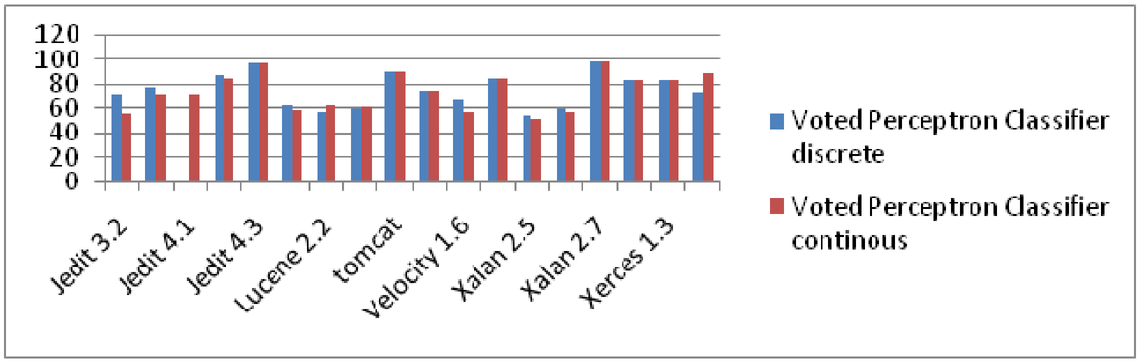

Figure 4: Represents accuracy comparison of Discrete and continuous data using Voted Perceptron classifier

In case of SOM, Figure 5 show almost similar accuracy output performance for both data set i.e discrete and continuous. But sometimes the accuracy is faced with the accuracy paradox. Consider a situation when TP =0, FP=0, TN=115,FN=15, this says that the classifier unable to predict TRUE class (TP=0), but can Classify FALSE class well (TN=115). In such a case, accuracy comes out to be 88%. To deal with this kind of situation, many other measures are consider to check the performance of classifier. Table 3 and Table 4 lists various other statistical measures like precision and RMSE. Precision is measure the percentage of actual correct samples out of all the examples labeled as positive by the classifier. Whereas RMSE is a measure of the correctness of classifier. It explains how distinct the classification model is from the actual prediction. RMSE is a good predictor of model about a Binary Classifier. RMSE varies from 0.0 to 1.0. and low values of RMSE for prediction of a good classifier model are desirable. Figure 5, Figure 6 and Figure 7 represents the RMSE value comparison for discrete and continuous data of the tthree classifiers: Naive Bayes, Voted Perceptron and SOM, used in the study.

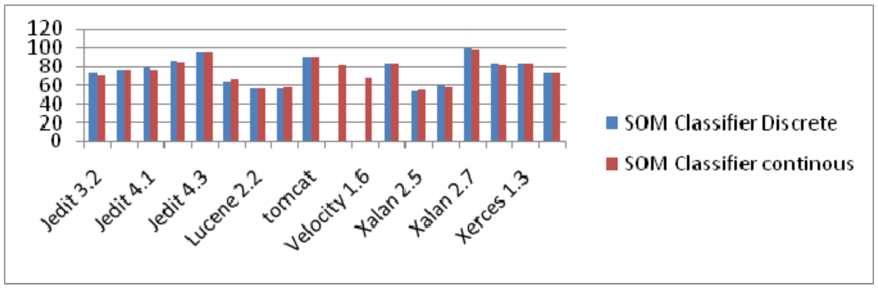

Figure 5: Represents accuracy comparison of Discrete and continuous data using SOM classifier

Table 3: Statistical measures of Naive Bayes Classifier for discretized (Dis.) and continuous (Conts.) features

Defect set	Data Type	Correctly classified	TP	FP	ROC	Precision	Recall	RMSE
Jedit 3.2.	Dis.	72.05	0.72	0.38	0.74	0.71	0.72	0.43
	Conts.	71.7	0.717	0.42	0.75	0.7	0.71	0.45
Jedit 4.0.	Dis.	77.45	0.775	0.586	0.57	0.74	0.77	0.42
	Conts.	78.4	0.78	0.54	0.68	0.76	0.78	0.43
Jedit 4.1.	Dis.	79.16	0.79	0.53	0.62	0.78	0.79	0.42
	Conts.	77.2	0.78	0.55	0.72	0.74	0.77	0.42
Jedit 4.2.	Dis.	86.6	0.87	0.57	0.63	0.85	0.87	0.33
	Conts.	86.3	0.86	0.65	0.72	0.84	0.86	0.35
Jedit 4.3.	Dis.	96.9	0.97	0.978	0.722	0.95	0.97	0.156
	Conts.	95.5	0.96	0.71	0.59	0.97	0.96	0.2
Lucene 2.0.	Dis.	63.5	0.63	0.38	0.63	0.67	0.67	0.48
	Conts.	69.7	0.697	0.32	0.755	0.71	0.7	0.33
Lucene 2.2.	Dis.	55	0.55	0.51	0.57	0.53	0.55	0.48
	Conts.	55.4	0.56	0.36	0.61	0.64	0.55	0.55
Lucene 2.4.	Dis.	63.2	0.63	0.41	0.61	0.62	0.63	0.48
	Conts.	57.6	0.58	0.32	0.68	0.69	0.57	0.59
Tomcat.	Dis.	90.7	0.91	0.91	0.74	0.82	0.91	0.28
	Conts.	87.8	0.88	0.62	0.78	0.88	0.88	0.33
Velocity 1.4.	Dis.	74.4	0.74	0.75	0.65	0.56	0.74	0.42
	Conts.	78	0.78	0.45	0.72	0.76	0.78	0.42
Velocity 1.6.	Dis.	64.1	0.64	0.53	0.597	0.61	0.64	0.47
	Conts.	67.2	0.672	0.54	0.65	0.64	0.67	0.54
Xalan 2.4.	Dis.	84.5	0.85	0.84	0.6	0.76	0.85	0.35
	Conts.	82.7	0.827	0.67	0.7	0.79	0.82	0.4
Xalan 2.5.	Dis.	54.6	0.55	0.47	0.56	0.55	0.55	0.5
	Conts.	56.1	0.56	0.46	0.59	0.79	0.56	0.6
Xalan 2.6.	Dis.	59.7	0.6	0.42	0.611	0.6	0.6	0.49
	Conts.	59.7	0.59	0.45	0.63	0.65	0.59	0.6
Xalan 2.7.	Dis.	98.7	0.99	0.99	0.56	0.97	0.98	0.02
	Conts.	61.8	0.61	0.27	0.71	0.98	0.61	0.58
Xerces 1.2.	Dis.	80.2	0.8	0.755	0.647	0.75	0.82	0.38
	Conts.	79	0.79	0.74	0.57	0.74	0.79	0.43
Xerces 1.3.	Dis.	83.2	0.83	0.74	0.62	0.78	0.83	0.37
	Conts.	81	0.81	0.66	0.59	0.79	0.81	0.41
Xerces 1.4.	Dis.	70.4	0.7	0.48	0.7	0.7	0.7	0.41
	Conts.	65.9	0.66	0.22	0.79	0.78	0.66	0.53

Table 4: Statistical measures of Voted Perceptron Classifier for discretized (Dis.) and continuous (Conts.) features

Defect set	Data Type	Correctly classified	TP	FP	ROC	Precision	Recall	RMSE
Jedit 3.2	Dis.	72.05	0.72	0.38	0.74	0.71	0.72	0.43
	Conts.	71.7	0.717	0.42	0.75	0.7	0.71	0.45
Jedit 4.0	Dis.	77.45	0.775	0.586	0.57	0.74	0.77	0.42
	Conts.	78.4	0.78	0.54	0.68	0.76	0.78	0.43
Jedit 4.1	Dis.	79.16	0.79	0.53	0.62	0.78	0.79	0.42
	Conts.	77.2	0.78	0.55	0.72	0.74	0.77	0.42
Jedit 4.2	Dis.	86.6	0.87	0.57	0.63	0.85	0.87	0.33
	Conts.	86.3	0.86	0.65	0.72	0.84	0.86	0.35
Jedit 4.3	Dis.	96.9	0.97	0.978	0.722	0.95	0.97	0.156
	Conts.	95.5	0.96	0.71	0.59	0.97	0.96	0.2
Lucene 2.0	Dis.	63.5	0.63	0.38	0.63	0.67	0.67	0.48
	Conts.	69.7	0.697	0.32	0.755	0.71	0.7	0.33
Lucene 2.2	Dis.	55	0.55	0.51	0.57	0.53	0.55	0.48
	Conts.	55.4	0.56	0.36	0.61	0.64	0.55	0.55
Lucene 2.4	Dis.	63.2	0.63	0.41	0.61	0.62	0.63	0.48
	Conts.	57.6	0.58	0.32	0.68	0.69	0.57	0.59
tomcat	Dis.	90.7	0.91	0.91	0.74	0.82	0.91	0.28
	Conts.	87.8	0.88	0.62	0.78	0.88	0.88	0.33
Velocity 1.4	Dis.	74.4	0.74	0.75	0.65	0.56	0.74	0.42
	Conts.	78	0.78	0.45	0.72	0.76	0.78	0.42
Velocity 1.6	Dis.	64.1	0.64	0.53	0.597	0.61	0.64	0.47
	Conts.	67.2	0.672	0.54	0.65	0.64	0.67	0.54
Xalan 2.4	Dis.	84.5	0.85	0.84	0.6	0.76	0.85	0.35
	Conts.	82.7	0.827	0.67	0.7	0.79	0.82	0.4
Xalan 2.5	Dis.	54.6	0.55	0.47	0.56	0.55	0.55	0.5
	Conts.	56.1	0.56	0.46	0.59	0.79	0.56	0.6
Xalan 2.6	Dis.	59.7	0.6	0.42	0.611	0.6	0.6	0.49
	Conts.	59.7	0.59	0.45	0.63	0.65	0.59	0.6
Xalan 2.7	Dis.	98.7	0.99	0.99	0.56	0.97	0.98	0.02
	Conts.	61.8	0.61	0.27	0.71	0.98	0.61	0.58
Xerces 1.2	Dis.	80.2	0.8	0.755	0.647	0.75	0.82	0.38
	Conts.	79	0.79	0.74	0.57	0.74	0.79	0.43
Xerces 1.3	Dis.	83.2	0.83	0.74	0.62	0.78	0.83	0.37
	Conts.	81	0.81	0.66	0.59	0.79	0.81	0.41
Xerces 1.4	Dis.	70.4	0.7	0.48	0.7	0.7	0.7	0.41
	Conts.	65.9	0.66	0.22	0.79	0.78	0.66	0.53

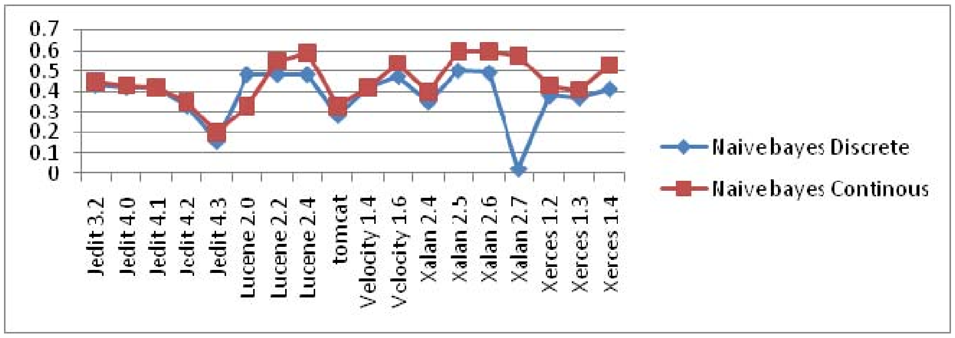

Figure 6: RMSE value comparison for Discrete and Continuous data for Naive Bayes

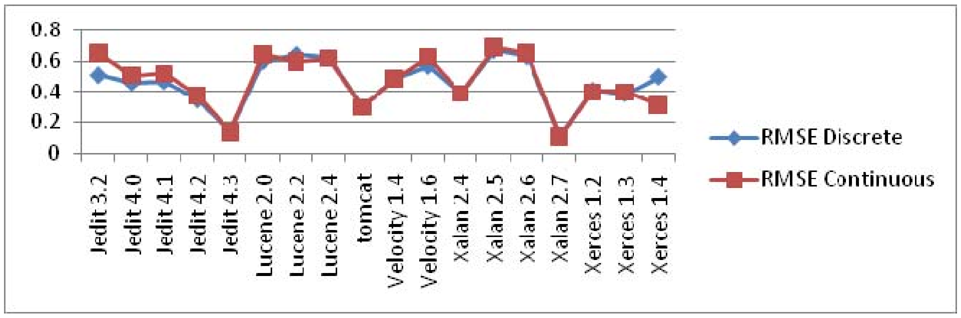

Figure 7: RMSE value comparison for Discrete and Continuous data for voted perceptron

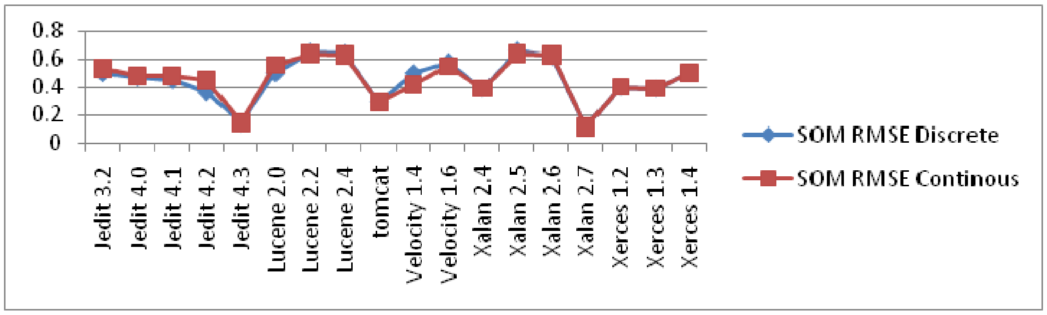

Figure 8: Represents RMSE value comparison for Discrete and Continuous data for SOM

Figure 6, generated using Naive Bayes classifier, indicates better RMSE curve for discretized data. Generally for all systems under study, it shows better classification and prediction model using the discretized data set. But for the Voted Perceptron (Figure 7), there is no remarkable difference in the RMSE curve using both discretized and continuous data sets. Similarly for SOM, in Figure 8, except for few values RMSE for both the data set shows similar output.

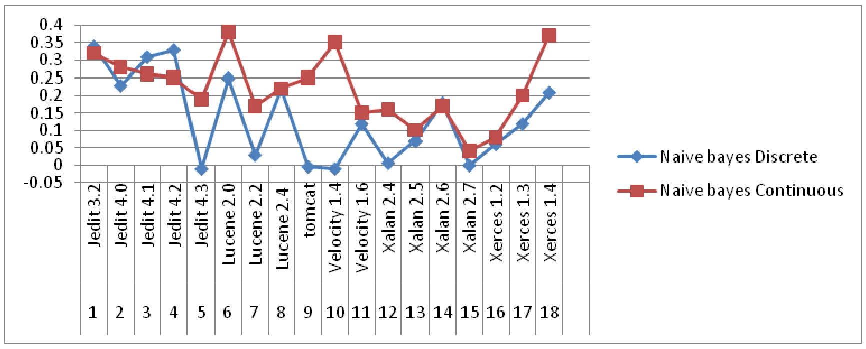

Figure 9: Kappa statistics for discrete and continuous data as result of the Naive Bayes Classifier

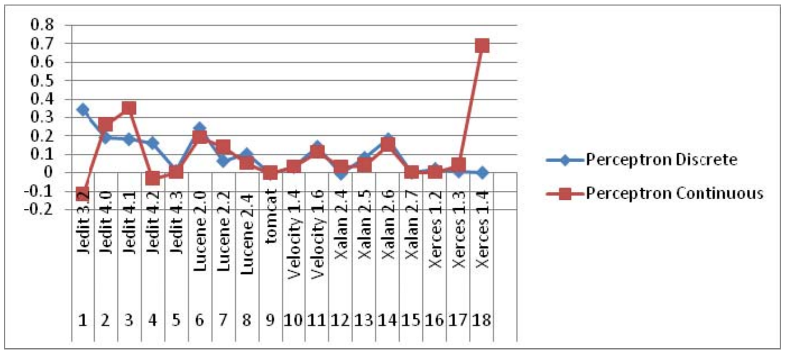

Figure 10: Kappa statistics for discrete and continuous data as result of the Voted Perceptron Classifier

Figure 10 show comparison of kappa statics for discrete and continuous data as a result of voted Perceptron classifier. The kappa curve for both discrete data and continuous is found to be similar except in two or three defect data sets like jedit 3.2, jedit 4.0, jedit 4.1and xerces 1.4.

7. CONCLUSION

This research work, has proposed a discretization method, in order to increase the overall efficiency of the existing CK Metrics prescribed for object oriented systems. The proposed discretization method is applied on, in total eighteen different systems taken from NASA repositories. During experimentation it is found that the threshold identified in the study in terms of MMV can be used to change the nature of continuous features into two classes {0,1}[15]. The discretization of values so preformed can help in increasing efficiency of classifiers like the Naive Bayes and the Voted Percepton in an significant manner [8]. The study suggests that by using certain threshold value of metrics, the efficiency of CK metrics in prediction of software quality can be increased towards predicting software defects at an early stages of its development. Out of eighteen data sets considered under experiment, the overall increase in accuracy after the proposed descretization, comes out in a range of 0.4% - 8%, if used for Naive Bayes and perceptron, which is most likely as both classifiers already have proven prediction capabilities. Majorly improvements can be seen but in case of support vector machine doesn't give prominent results. Future dimension of the proposed research work could be identifying ensemble classifiers or inclusion of genetic algorithm, towards direction of building of novel methods for gaining more prompt predictive capabilities. Genetic algorithms can be more helpful in optimizing the threshold parameters used for discretization. The threshold values are dependent on various factors related to software development environment and the software quality requirement of the system, so dynamic adoptability of these parameters could be the one of the most deciding factors.

REFRENCES:

[1] Deepak Arora, Pooja Khanna and AlpikaTripathi, Shipra Sharma and Sanchika Shukla, Software Quality Estimation through Object Oriented Design Metrics, IJCSNS International Journal of Computer Science and Network Security, Vol.11 NO.4, April 2011.

[2] Zimmermann, T., Premraj, R., & Zeller, A. (2007, May). Predicting defects for eclipse. In Proceedings of the third international workshop on predictor models in software engineering (p. 9). IEEE Computer Society.

[3] Jiang Y, Cukic B, Menzies T. Fault prediction using early lifecycle data. In: 18th IEEE International Symposium on Software Reliability.Trollhattan; 2007. p. 237-246.

[4] Emam KE, Melo W, Machado JC. The prediction of faulty classes using object-oriented design metrics. Journal of Systems and Software2001; 56:63-75.

[5] Sandhu PS, Brar AS, Goel R, Kaur J, Anand S. A model for early prediction of faults in software systems. In: 2nd International Conference on Computer and Automation Engineering. Singapore; 2010. p. 281-285.

[6] Jiang Y, Cukic B, Menzies T. Cost curve evaluation of fault prediction models. In: 19th International Symposium on Software

Reliability Engineering. Seattle; 2008. p. 197-206.

[7] Jiang Y, Lin J, Cukic B, Menzies, T. Variance analysis in software fault prediction models. In: 20th International Symposium on Software Reliability Engineering. Mysuru; 2009. p. 99-108.

[8] Menzies T, Greenwald J, Frank A. Data mining static code attributes to learn defect predictors. IEEE Transactions on Software Engineering2007; 33:2-13. 22.

[9] Chidamber, Shyam , Kemerer, Chris F. "A Metrics Suite for Object-Oriented Design."M.I.T. Sloan School of Management E53-315, 1993.

[10] A Systematic Literature Review of Software Defect Prediction: Research Trends, Datasets, Methods and Frameworks,JournalOfSoftwareEngineering, April2015.

[11] Ishani Arora a, Vivek Tetarwala, Anju Sahaa , Open issues in software defect prediction, Elsevier, 2015

[12] Ahmed H. Yousef, Extracting software static defect models using data minig, Elsevier , 2015.

[13] Wang, Shuo, and Xin Yao. "Using class imbalance learning for software defect prediction." IEEE Transactions on Reliability 62.2 (2013): 434-443.

[14] Ren, J., Qin, K., Ma, Y., & Luo, G. (2014). On software defect prediction using machine learning. Journal of Applied Mathematics, 2014.

[15] P. Kapoor, D.Arora, Ashwani, Effects of Mean Metric Value Over CK Metrics Distribution Towards Improved Software Fault Predictions, proc Springer's International Conference IC4S 2016.

[16] Rish, Irina. "An empirical study of the naive Bayes classifier." IJCAI 2001 workshop on empirical methods in artificial intelligence. Vol. 3. No. 22. IBM New York, 2001.

[17] Garcia, Salvador, et al. "A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning." IEEE Transactions on Knowledge and Data Engineering 25.4 (2013): 734-750.S.

[18] Kotsiantis, D. Kanellopoulos Discretization techniques: A recent survey GESTS Int. Trans. Computer Sci. Eng., 32 (2006), pp. 47–58

[19] A.D. Bakar, A. Sultan, H. Zulzalil and J. Din, 2014. Predicting Maintainability of Object-oriented Software Using Metric Threshold. Information Technology Journal, 13: 1540-1547.

[20] Metz, CE (October 1978). "Basic principles of ROC analysis" (PDF). Semin Nucl Med. 8 (4): 283–98.

[21] Kaya, Fatih. "Discretizing continuous features for naïve Bayes and C4. 5 classifiers." University of Maryland publications: College Park, MD, USA (2008).

[22] Kohavi, Ron, and Mehran Sahami. "Error-Based and Entropy-Based Discretization of Continuous Features." KDD. 1996.

[23] Rish, Irina. "An empirical study of the naive Bayes classifier." IJCAI 2001 workshop on empirical methods in artificial intelligence. Vol. 3. No. 22. IBM New York, 2001.

[24] http://promisedata.googlecode.com

[25] P. Singh and S. Verma, "An Investigation of the Effect of Discretization on Defect Prediction Using Static Measures," 2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies, Trivandrum, Kerala, 2009, pp. 837-839.