Scientific Online

ARTIFICIAL INTELLIGENCE AND APPLICATIONS
ISSN(Print): 2374-4979 ISSN(Online): 2374-4987
In Press

ARTIFICIAL INTELLIGENCE AND APPLICATIONS

Cluster Ensemble and Probabilistic Neural Network Modeling of Class Imbalance Learning in Software Defect Prediction

Biprodip Pal*, Abu Hasan, Mumu Aktar, Nazmul Shahdat

Department of Computer Science & Engineering, Rajshahi University of Engineering & Technology, Bangladesh
***Corresponding author**: biprodip.cse@gmail.com

Abstract:
Machine learning techniques are frequent for the complicated task of predicting software defects. Often the prediction models fail to predict defects successfully as the between class imbalance increases in the training data. This paper conducts experiments on NASA KC1 dataset. Experiments were carried out on five different datasets with different class imbalance ratio, made from original KC1 dataset. Performance variations of each classifier were analyzed to detect software defects as the imbalance ratio increases for training datasets. Moreover different classifier's performances were also investigated while trained with same datasets to find out the better one for each case. It is found that our proposed Cluster Ensemble based Probabilistic Neural Network approach has performed better than all other approaches including SMOTE based Probabilistic Neural Network for the given data sets. It is also found that incorporating Cluster Ensemble with SMOTE based Probabilistic Neural Network approach performs better than only SMOTE based Probabilistic Neural Network approach.

Keywords:
Class Imbalance; Cluster Ensemble; Software Defect; Probabilistic Neural Network; Machine Learning;

1. INTRODUCTION

Software development consists of a set of tasks like requirement analysis, validation and evolution. In the evolution process bug or defect fixing is an important part. Defects in a software product prevent a it from meeting software requirement or end-user expectations. Usually software's are prone to many defects like Arithmetic, Logic, Syntax, Team working defects *etc*. Software defect identification and fixing is often very complex asking for time consuming efforts. Since many software play a vital role in our daily life events as well as in sophisticated areas of countries, defects can cause problems, ranging from minor to catastrophic glitches leading to loss of life [1]. Thus complex time-consuming problem of defect prediction must be efficiently solved, leading to extensive research activities in this area.

Several attempts have been taken to use data mining and machine learning techniques for intelligent software defect prediction. Few researches took into consideration the total Lines of Code (LOC)[2] as a parameter for defect occurance, since it represents the complexity. Cyclomatic complexity metric and Halstead complexity metrics [3, 4] are also proposed as important parameters by others. Besides, other research used machine learning techniques such

as Munson *et al*. proposed classification model [5], Shen *et al*. proposed linear regression model [6] for software defect prediction. Models having less historical data asks for Just-in-time (JIT) defect prediction or in other words defect prediction model for changes proposed by Mockus *et al*. [7]. Research on Transfer learning approaches has come out with efficient cross-project defect prediction models [8, 9] working well for data in new domain. Li *et al* [10] suggested a sample-based process that selects and tests a small proportion of modules to build a model for a large software system to predict defects. Song *et al* [11] suggested a general software defect prediction framework evaluating prediction performance of opposing schemes for particular historical data sets. Models are constructed based on evaluated knowledge that predicts software defects from new data.

The performance of machine learning and data mining techniques depends on the nature of available data. Significant decrease of the performance of most standard learning algorithms can be caused by imbalanced data. This work represents Cluster Ensemble (CE) based data preparation for Probabilistic Neural Network (PNN) to deal with the imbalance software defect prediction data. It also includes analyzing the well-known Synthetic Minority Oversampling Technique (SMOTE) for imbalanced data with PNN. Then the effect of applying CE over SMOTE processing was analyzed for PNN based prediction of defected software. The experiment was evaluated using KC1 dataset, created by NASA**.**

The desire of this work is to identify defects efficiently using machine learning techniques for imbalanced training datasets of existing software metrics which leads to reducing the software developing and maintenance cost.

2. CLASS IMBALANCE PROBLEM AND RATIO

Imbalanced data sets consist of an unequal class distribution. In common understanding class imbalance or between-class imbalance corresponds to major, and frequently excessive imbalances. Often one class may out represent another class with an order of 100:1, 1,000:1, and 10,000:1 and so on [12–15]. A typical scenario is illustrated in **Figure 1**.

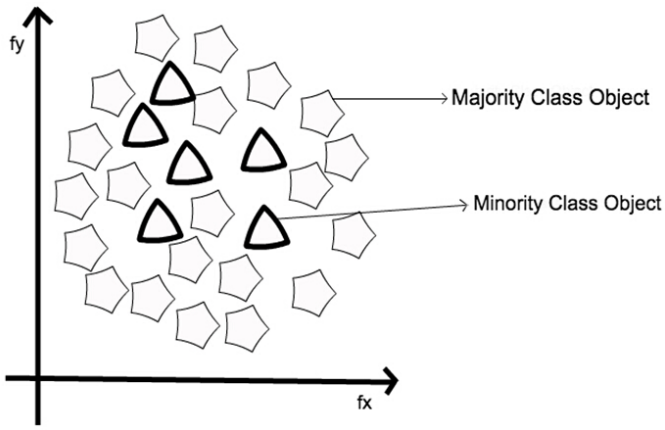

Figure 1. Class imbalance in datasets, triangle samples are out represented by pentagonal objects.

For imbalance problems the Imbalance Ratio (IR) is calculated as equation 1.

$$IR = \frac{T_{maj}}{T_{\min}} \tag{1}$$

Where,

T_{maj}= Total instances of majority class and

T_{min}= Total instances of minority class

Imbalance in data often results from problems that are inherently based on imbalanced data. For example, if a data set is considered that consists of fraud transaction samples, naturally normal transaction samples will greatly exceed fraudulent samples. On the other hand when data collection process is limited, class imbalances occur artificially in some domains that doesn't have intrinsic imbalance in data. Again in certain cases, the data abounds and the scientist decides the quantity of samples to be selected [16]. In imbalanced cases classifiers used to provide an imbalanced degree of predictive accuracy with the majority class having better percentage of accuracy while the minority class having very poor accuracies. But a balanced accuracy (ideally 100 percent) for both of the classes on the data set is preferable [17].

3. METHODOLOGY

3.1 Cluster Ensemble

Cluster Ensemble (CE) aggregates cluster instances based on different subset of features. The general goal of clustering is to partition a set of data such that "similar" data points are grouped into the same cluster while those from different clusters are grouped as "dissimilar." Thus it attempts to increase interclass distance and decrease intraclass distance. Given a feature space A = {1,2,. . . , p} as the set of indices of coordinates in $\Re^p$ and n observations {X_1, . . . , X_n} $\in \Re^p$, each clustering vector ν is a subset of A. Each clustering vector is expanded iteratively based on quality measure Q which is calculated from equation 2.

$$Q(\nu\prime) = SS_W(\nu\prime)/SS_B(\nu\prime) \tag{2}$$

Where,

SS_W is within-cluster sum of squared distances

SS_B is between-cluster sum of squared distances

Both SS_W and SS_B are computed on currently used feature subset ($\nu\prime$) , respectively.

τ= Number of consecutive unsuccessful attempts,

τ_m = Maximal allowed value of τ in expanding a clustering vector.

The growth of a clustering vector is described in Algorithm 1 [18].

Algorithm 1. The growth of a clustering vector $\nu\prime$

1: Set $\tau = 0$ and $\nu\prime$={NULL};

2: Initially select $\nu\prime \leftarrow (\nu_1^{(0)}, \ldots, \nu_b^{(0)})$ using feature contest;

3: repeat

4: Sample $\nu_1, \ldots, \nu_b$, from the feature space A where s<p ;

5: Projected the data on feature space ($\nu\prime$, $\nu_{1,\ldots,}\nu_b$) and Apply K-means Clustering;

6: if $Q(\nu\prime, \nu_{1,\ldots,}\nu_b) > Q(\nu\prime)$ then

7: reject $\{\nu_{1,\ldots,}\nu_b\}$ and set $\tau = \tau+1$;

8: else

9: expand $v\prime$ by ($v\prime$, $v_{1,\ldots,}v_b$) and set τ= 0;

10: end if

11: until $\tau = \tau_m$;

Algorithm 2. Feature contest

1: Set i = 1;

2: repeat

3: Sample $\{v_1^{(i)},\ldots, v_b^{(i)}\} \in$ A;

4: Apply K-means to the data projected on ($v_1^{(i)},\ldots, v_b^{(i)}$) to get Q ($v_1^{(i)},\ldots, v_b^{(i)}$);

5: Set i=i+1;

6: until i=q;

7: Set ($v_1^{(0)},\ldots, v_b^{(0)}$) as min Q ($v_1^{(i)},\ldots, v_b^{(i)}$) of q subsets;

Feature contest is desired to present a good initialization for the clustering vector growth and the procedure is illustrated in Algorithm 2. When there are large numbers of noisy features, the above process comes out with resultant clustering vector consisting of discriminating features that lead to a "good" clustering.

3.2 Probabilistic Neural Network

A Probabilistic Neural Network (PNN) is a form of statistical discriminant analysis process where operations are designed into a multilayered feedforward network as given in **Figure 2**. PNN replaces the sigmoid activation function of neural networks with an exponential function [19].

PNN is fast, based on Bayesian classification and guaranteed converging to optimal classifier when the representative training data is large enough. PNN is supervised inherently parallel structure with no local minima issues. It allows one-shot classification thus learning processes have no relation with recalling processes.

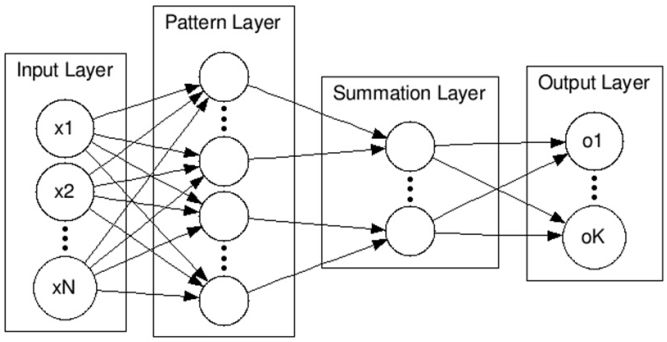

Figure 2. Probabilistic Neural Network Architecture

PNN consists of a pattern layer or radial basis layer where radial basis function works as the transfer function and can be calculated by-

$$h(x) = \phi(||x - x_c||) \tag{3}$$

Where,

ø is the radial basis function

x is the input of the neuron

x_c is the centre of the neuron

|| is usually used for Euclidean distance

The layer, computes the probabilistic density function (PDF) of the input training samples according to their distribution values . Modifying simple PDF and applying to the input forms output vector P of the Pattern Layer in the PNN as given in equation 4 [20].

$$P_h = \exp\left(\frac{-\sum_i (X_i - W_{ih}^{xh})^2}{2\sigma^2}\right) \tag{4}$$

The Summation Layer performing weighted sum operation of it's inputs and is defined as

$$net_j = \frac{1}{N_j}\sum_h W_{hj}^{hy}.P_h \quad and \quad N_j = \sum_h W_{hj}^{hy} \tag{5}$$

Output Layer performs a max operation to select the output class value. The inference procedure for output vector O is

$$net_j = \max_k (net_k) \text{ then } O_j=1 \text{ else } O_j= 0$$

Where,

i is the number of input layers

h is the number of hidden layers

j is the number of output layers

k is the number of training examples

N_k is the number of classification

σ is the smoothing parameters (standard deviation), $0.1 < \sigma < 1$

X is the input vector

W_{ih}^{xh} is the connection weight between the input layer X and the pattern layer P

W_{hj}^{hy} is the connection weight between the pattern layer P and the summation layer Y

Normally spread factor σ value is taken as 0.1, 0.05 or 0.01. Too small value leads to neurons with less input space covering capability thus poor generalization while too large σ gives values near 1.0 for all inputs [20].

3.3 SMOTE

Synthetic minority oversampling technique (SMOTE) has become successful in several applications**.** Instead of replacement with over-sampled data, minority class is over-sampled with 'synthetic' examples [17].

Synthetic data is created based on the feature space similarities between existing minority examples. For minority class subset $S_{min} \in S$, K-nearest neighbors are considered for each sample $x_i \in S_{min}$, where K is an integer. One of the K-nearest neighbors are selected at random and the difference between original and selected neighbor is calculated. Then the feature vector difference is multiplied with a random number between [0,1]. Finally, new sample x_{new} is generated using equation (6).

$$x_{new} = x_i + (x_i\prime - x_i) \times \delta \quad (6)$$

Where $x_i \in S_{min}$ is the minority instance under consideration, $x_i\prime$ is selected K-nearest neighbor of

x_i: $x_i\prime \in S_{min}$, and $\delta \in [0,1]$ is a random number. Therefore, the resulting synthetic instance according to (6) is a point along the line segment joining x_i under consideration and the randomly selected K-nearest neighbor $\mathbf{x}_i$ [12] [17]. The over all process is given in Algorithm 3 as pseudo code.

Algorithm 3. SMOTE

1. For each data point $x_i \in S_{min}$
2. Select K Nearest Neighbors $\{x_1\prime, x_2\prime, \ldots, x_k\prime\}$ from the entire dataset S
3. Repeat
4. Randomly pick a neighbor $x_i\prime \in S_{min}$
5. Calculate the distance d= ($x_i\prime$ - x_i)
6. Randomly choose a resample point x_{new} as $x_{new} = x_i + d \times \delta$
7. Until total oversampling is completed

The synthetic examples cause the classifier to create larger and less specific decision regions. These synthetic samples help break the ties introduced by simple oversampling, and furthermore, augment the original data set in a manner that generally significantly improves learning.

4. PROPOSED APPROACH

SMOTE adds to the dataset a set of new instances that are supposed to come out with better learning space for classifier models. But very often there remains a possibility of noisy features in the data sets. Besides, taking a lot of features lead to over fitting the classifier model. Generating unreal samples consisting of the same set of features as that of the original samples therefore increases instances that also have noisy features.

So, CE is proposed to select appropriate set of features and then feed the projected data into the classifier. As CE reduces noisy features, instances become more suitable for learning. Instead of generating extra samples, noisy feature reduction helps the classifiers to learn better about the classes from existing data. Again as the CE approach also select discriminative features out of all, distinguishing characteristics of the classes are used to generate classifier model that results better classification. Moreover the projected data on the selected features have reduced dimension so, faster processing is ensured. Block diagram of the proposed processing approach is illustrated in **Figure 3**.

5. DATASET & EXPERIMENTAL SETUP

5.1 Dataset

KC1 dataset is from a NASA Metrics Data Program [21]. Software engineering model verification, and improvement experiments uses it widely. KC1 contain static code measures like includes McCabe, Halstead features, LOC *etc*. KC1 is a C++ coded system for storage management for receipt and ground data processing [21, 22]. The data set contains 21 software product metrics as listed in **Table 1**.

The test dataset with 703 instances and train dataset with 1406 candidates were built from the original dataset.

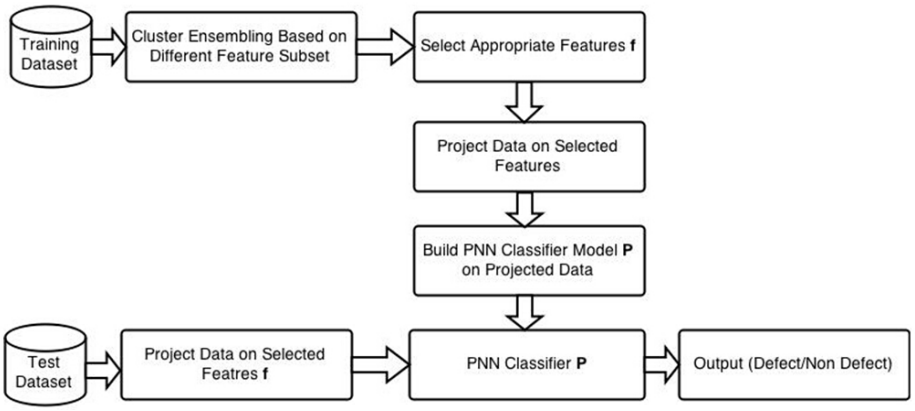

Figure 3. Block diagram of proposed processing approach.

Table 1. Original KC1 Dataset Description

Dataset	Attribute Names	Description	Classes	Size
KC1	loc	McCabe's line count of code	2	2109
	v(g)	McCabe "cyclomatic complexity"		
	ev(g)	McCabe "essential complexity"		
	iv(g)	McCabe "design complexity"		
	n	Halstead total operators + operands		
	v	Halstead "volume"		
	l	Halstead "program length"		
	d	Halstead "difficulty"		
	i	Halstead "intelligence"		
	e	Halstead "effort"		
	b	Halstead		
	t	Halstead's time estimator		
	lOCode	Halstead's line count		
	lOComment	Halstead's count of lines of comments		
	lOBlank	Halstead's count of blank lines		
	lOCodeAndComment	numeric		
	uniq_Op	unique operators		
	uniq_Opnd	unique operands		
	Op	total operators		
	total_Opnd	total operands		
	branchCount	of the flow graph		
	problems	module has/has not one or more reported defects		

Than a total of 5 dataset, each with 1200 patterns were selected from the 1406 instances of the full train dataset. These 5 dataset has different IR values. The KC1 dataset comes out with an IR of 4, and this experiment were carried out for up to three times the default IR value. The dataset information is given in **Table 2**.

Table 2. Training Dataset Description

Datasets	No. of instances		IR (max/min)
	Class1 (True)	Class2 (False)	
D1	240	960	4
D2	171	1029	6
D3	133	1067	8
D4	109	1091	10
D5	92	1108	12
T1	45	658	-

5.2 Evaluation Metric

Informative assessment metrics like Confusion Matrix and Receiver Operating Characteristics (ROC) curves are considered for evaluation since singular assessment criteria like the overall accuracy or error rate, does not offer enough information to evaluate imbalanced learning.

5.3 Confusion Matrix

Confusion matrix visualizes an algorithm's performance. Usually it is used for supervised learning algorithms. It contains information about actual and hypothesis output done by a classification system.

In a basic binary classification problem, let $\{p; n\}$ be the true positive and negative class label and $\{C1;C2\}$ are the predicted positive and negative class labels. **Table 3** illustrates classification performance formulation by a confusion matrix [12].

Table 3. Sample Confusion Matrix

	p (Target)	n (Target)
C1 (Predicted)	TP	FP
C2 (Predicted)	FN	TN
	PC	NC

In this paper, we use the minority class (Class 1) as the positive class and the majority class(Class 2) as the negative class. Thus accuracy and error rate are defined as

$$Accuracy = (TP + TN)/(PC + NC) \tag{7}$$

$$ErrorRate = 1 - accuracy \tag{8}$$

5.4 Receiver Operating Characteristics (ROC) Curves

The ROC curve plots two evaluation matrices namely the True Positive Rate (Sensitivity) in function of the False Positive Rate for different cut-off points of a parameter. True Positives Rate (TPR) and False Positives Rate (FPR), are defined as:

$$TPR = \frac{TP}{PC} \tag{9}$$

$$FPR = \frac{FP}{NC} \tag{10}$$

TPR/FPR pair's corresponding to a particular decision threshold are represented at each point on the ROC curve. Thus any of these points correspond to the performance of a single classifier for a given distribution. Thus the visual representation of the relative trade-offs between the benefits (TPR) and costs (FPR) are easily understood. The Area Under the ROC curve (AUC) is a measure of how well a parameter can distinguish between given classes. It corresponds to the probability of identifying one of the two instances correctly [23–25].

6. RESULTS AND DISCUSSION

6.1 Performance Analysis of PNN

Firstly, the performances of the PNN approach for classification of each of the five dataset were evaluated. **Table 4** - **Table 8** depicts the confusion matrix of PNN performance for training with D1 to D5 respectively and testing on the T1 test dataset. The overall accuracy increased gradually from 61.9 at D1 to 64.4 at D5.

Table 4. Confusion Matrix of PNN based Classification for Training with D1.

	Class 1 (Target)	Class 2 (Target)	Accuracy (%)
Class 1 (Predicted)	40	263	13.2
Class 2 (Predicted)	5	395	98.8
Accuracy (%)	88.9	60.0	61.9

Table 5. Confusion Matrix of PNN based Classification for Training with D2.

	Class 1 (Target)	Class 2 (Target)	Accuracy (%)
Class 1 (Predicted)	39	248	13.6
Class 2 (Predicted)	6	410	98.6
Accuracy (%)	86.7	62.3	63.9

Figure 4 illustrates the ROC analysis of the corresponding performances. As the samples for the majority class increases with IR, performances become almost same for the PNN. Thus the classification of D1 training is poor in contrast to the other 4 datasets.

Table 6. Confusion Matrix of PNN based Classification for Training with D3.

	Class 1 (Target)	Class 2 (Target)	Accuracy(%)
Class 1 (Predicted)	38	247	13.3
Class 2 (Predicted)	7	411	98.3
Accuracy(%)	84.4	62.5	63.9

Table 7. Confusion Matrix of PNN based Classification for Training with D4.

	Class 1 (Target)	Class 2 (Target)	Accuracy(%)
Class 1 (Predicted)	38	246	13.4
Class 2 (Predicted)	7	412	98.3
Accuracy(%)	84.4	62.6	64.0

Table 8. Confusion Matrix of PNN based Classification for Training with D5.

	Class 1 (Target)	Class 2 (Target)	Accuracy(%)
Class 1 (Predicted)	38	243	13.5
Class 2 (Predicted)	7	415	98.3
Accuracy(%)	84.4	63.1	64.4

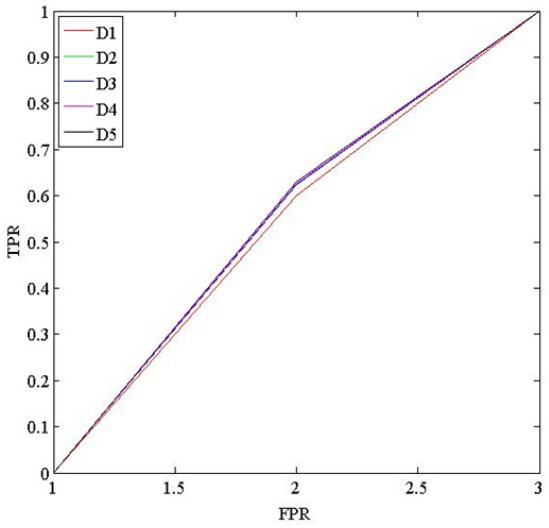

Figure 4. ROC curve of PNN.

6.2 Performance of CE based PNN (CEPNN) Approach

Secondly, CE based PNN approach classification performance was evaluated for D1 to D5. **Table 9** - **Table 13** illustrates the corresponding confusion matrix. Now, the overall accuracy increased gradually from 78.1 for training with D1 to 82.1 for D5. It is clear that CE provides better performance compared to PNN only approach. The ROC plot in **Figure 5** shows that the D1 training performance is inferior to others.

Table 9. Confusion Matrix of CEPNN based Classification for Training with D1.

	Class 1 (Target)	Class 2 (Target)	Accuracy(%)
Class 1 (Predicted)	28	137	17.0
Class 2 (Predicted)	17	521	96.8
Accuracy(%)	62.2	79.2	78.1

Table 10. Confusion Matrix of CEPNN based Classification for Training with D2.

	Class 1 (Target)	Class 2 (Target)	Accuracy(%)
Class 1 (Predicted)	24	110	17.9
Class 2 (Predicted)	21	548	96.3
Accuracy(%)	53.3	83.3	81.4

Table 11. Confusion Matrix of CEPNN based Classification for Training with D3.

	Class 1 (Target)	Class 2 (Target)	Accuracy(%)
Class 1 (Predicted)	24	108	18.2
Class 2 (Predicted)	21	550	96.3
Accuracy(%)	53.3	83.6	81.7

Table 12. Confusion Matrix of CEPNN based Classification for Training with D4.

	Class 1 (Target)	Class 2 (Target)	Accuracy(%)
Class 1 (Predicted)	24	106	18.5
Class 2 (Predicted)	21	552	96.3
Accuracy(%)	53.3	83.9	81.9

6.3 Performance of PNN on SMOTE (SMOTEPNN) based Processed Data

Table 13. Confusion Matrix of CEPNN based Classification for Training with D5.

	Class 1 (Target)	Class 2 (Target)	Accuracy(%)
Class 1 (Predicted)	24	105	18.6
Class 2 (Predicted)	21	553	96.3
Accuracy(%)	53.3	84.0	82.1

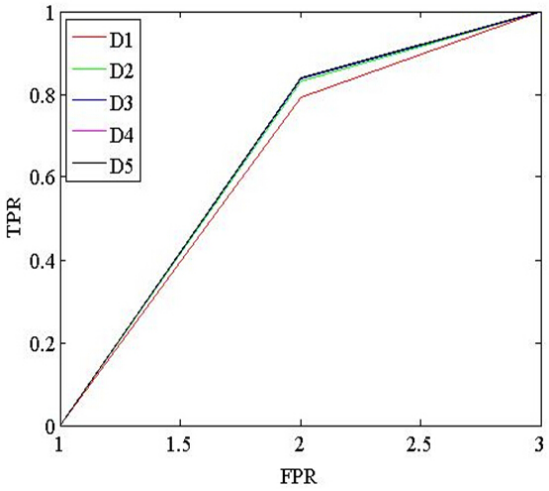

Figure 5. ROC Curve of CE PNN.

Table 14 to **Table 18** analyzes the performance of SMOTEPNN approach to classify the test dataset based on training with D1 to D5 respectively. The confusion matrices here, also demonstrates that increment in the IR comes out with better classification of the majority class but with a cost of decreasing minority class classification. It is to be noted that the accuracy of the majority sample classification is poor enough compared to that of the CEPNN approach. Thus a lot of samples are misclassified as a result the overall accuracy decreases. **Figure 6** shows that only D5 training has given a comparable reliable difference in evaluation in terms of the ROC plot with others.

Table 14. Confusion Matrix of SMOTEPNN based Classification for Training with D1.

	Class 1 (Target)	Class 2 (Target)	Accuracy(%)
Class 1 (Predicted)	41	306	11.8
Class 2 (Predicted)	4	352	98.9
Accuracy(%)	91.1	53.5	55.9

6.4 Performance of PNN on CE and SMOTE (CESMOTEPNN) based Processed Data

Finally the CESMOTEPNN performances for classification were evaluated. The confusion matrices in the **Table 19** to **Table 23** illustrates the rise in the corresponding overall accuracy for training with D1 to D5 compared to that

Table 15. Confusion Matrix of SMOTEPNN based Classification for Training with D2.

	Class 1 (Target)	Class 2 (Target)	Accuracy(%)
Class 1 (Predicted)	41	293	12.3
Class 2 (Predicted)	4	365	98.9
Accuracy(%)	91.1	55.5	57.8

Table 16. Confusion Matrix of SMOTEPNN based Classification for Training with D3.

	Class 1 (Target)	Class 2 (Target)	Accuracy(%)
Class 1 (Predicted)	39	289	11.9
Class 2 (Predicted)	6	369	98.4
Accuracy(%)	86.7	56.1	58.0

Table 17. Confusion Matrix of SMOTEPNN based Classification for Training with D4.

	Class 1 (Target)	Class 2 (Target)	Accuracy(%)
Class 1 (Predicted)	39	277	12.3
Class 2 (Predicted)	6	381	98.4
Accuracy(%)	86.7	57.9	59.7

Table 18. Confusion Matrix of SMOTEPNN based Classification for Training with D5.

	Class 1 (Target)	Class 2 (Target)	Accuracy(%)
Class 1 (Predicted)	38	258	12.8
Class 2 (Predicted)	7	400	98.3
Accuracy(%)	84.4	60.8	62.3

of SMOTEPNN due to incorporating CE with SMOTEPNN. **Figure 7** depicts that D5 training has the superior curve as like previous approaches and also the D1 training has less promising curve but they are better than corresponding SMOTEPNN in terms of ROC analysis.

6.5 ROC plot analysis of All Approaches for Classification

To find the contrast within all the classification approaches for training with same IR data, the ROC curve is plotted classifying T1 when trained with D1 (IR=4) has been plotted in **Figure 8**. Similarly ROC curve while trained with each of the train dataset D2, D3, D4, D5 has been plotted in **Figure 9** to **Figure 12** respectively.

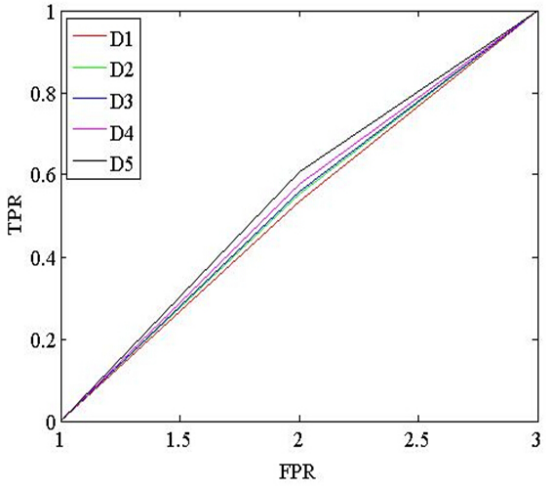

Figure 6. ROC Curve of SMOTE PNN.

Table 19. Confusion Matrix of CESMOTEPNN based Classification for Training with D1.

	Class 1 (Target)	Class 2 (Target)	Accuracy(%)
Class 1 (Predicted)	35	208	14.4
Class 2 (Predicted)	10	450	97.8
Accuracy(%)	77.8	68.4	69.0

Table 20. Confusion Matrix of CESMOTEPNN based Classification for Training with D2.

	Class 1 (Target)	Class 2 (Target)	Accuracy(%)
Class 1 (Predicted)	31	171	15.3
Class 2 (Predicted)	14	487	97.2
Accuracy(%)	68.9	74.0	73.7

Table 21. Confusion Matrix of CESMOTEPNN based Classification for Training with D3.

	Class 1 (Target)	Class 2 (Target)	Accuracy(%)
Class 1 (Predicted)	31	170	15.4
Class 2 (Predicted)	14	488	97.2
Accuracy(%)	68.9	74.2	73.8

It should be noted that, in all cases CEPNN performs best. The CEPNN curve has always come out with promising slope that leads to more true positive classification. As the IR increased from 4 for D1 to the CEPNN ROC curve tend to move upwards ensuring more true positive rate and less false positive rate. As can be seen form the figures,

Table 22. Confusion Matrix of CESMOTEPNN based Classification for Training with D4.

	Class 1 (Target)	Class 2 (Target)	Accuracy(%)
Class 1 (Predicted)	31	160	16.2
Class 2 (Predicted)	14	498	97.3
Accuracy(%)	68.9	75.7	75.2

Table 23. Confusion Matrix of CESMOTEPNN based Classification for Training with D5.

	Class 1 (Target)	Class 2 (Target)	Accuracy(%)
Class 1 (Predicted)	30	148	16.9
Class 2 (Predicted)	15	510	97.1
Accuracy(%)	66.7	77.5	76.8

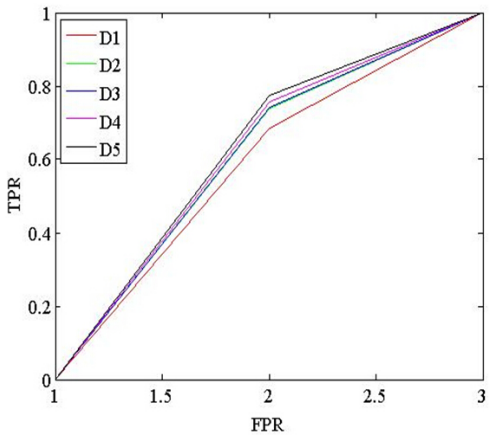

Figure 7. ROC Curve of CF SMOTE PNN.

the CE based approaches performed better than others. Since, next to the CEPNN approach, CESMOTEPNN can be selected as a good classification approach.

On the other hand as illustrated in figures the SMOTEPNN without CE is not used, performs as a random classifier that is not reliable at all. Besides, with increasing IR of training datasets, the distance between PNN and SMOTEPNN declines thus both of the classifiers are unsuited for classification. It's clear from the figures that as IR value grows for training data, there are no significant change in the ROC plot of SMOTEPNN and it is always bellow the other curves.

7. CONCLUSION

In this paper a cluster ensemble based probabilistic neural network approach is proposed for imbalanced data classification. It is found that incorporating CE approach with the PNN comes out with much reliable classification compared to that of state of the art SMOTE technique based PNN approach. The approach is also superior to PNN

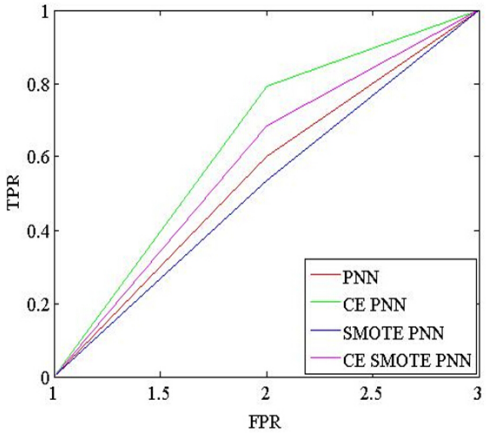

Figure 8. ROC of D1 training for All Techniques.

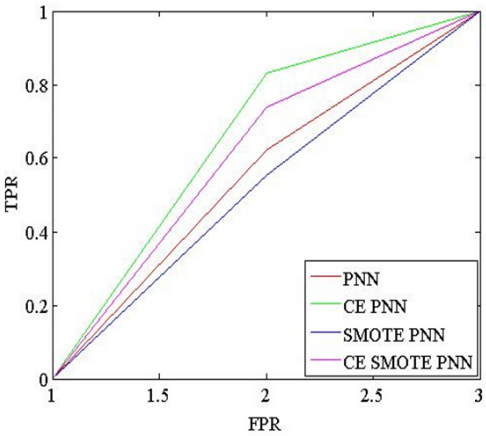

Figure 9. ROC of D2 training for All Techniques.

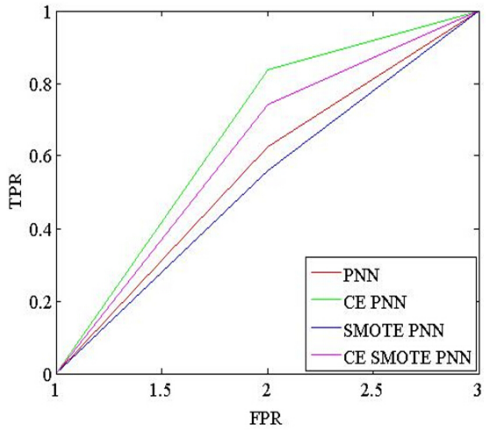

Figure 10. ROC of D3 training for All Techniques.

only classification for same training and test environment. Furthermore, the output demonstrates the superiority of

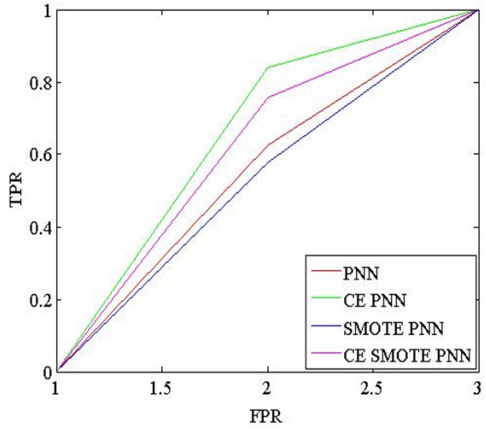

Figure 11. ROC of D4 training for All Techniques.

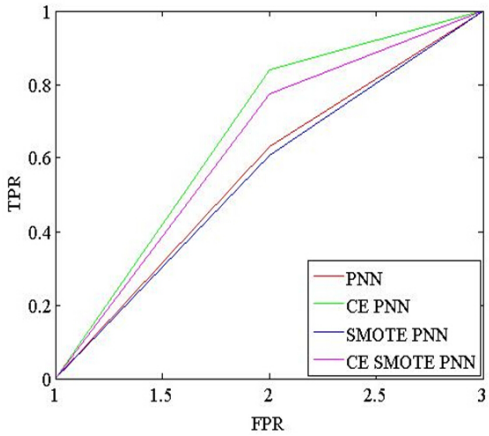

Figure 12. ROC of D5 training for All Techniques.

CE with SMOTE based PNN as it performs better than only SMOTE based PNN classifier. Future works can be carried out to check the impact of CE on other classifier performances and combined CE and other feature reduction approaches like PCA to reduce dimension for more classification from imbalanced data. Other important attempts can be improvement of the minority class classification accuracy through automatic adjustment of parameters that will lead to an adaptive CE approach.

References

[1] I. Sommerville, *Software Engineering 9th Edition*. Pearson Addison-Wesley, 2010.

[2] F. Akiyama, "An Example of Software System Debugging," in *Proceedings of the International Federation of Information Processing Societies Congress*, vol. 71, pp. 353–359, 1971.

[3] M. H. Halstead *et al.*, "Elements of Software Science (Operating and Programming Systems Series)," *New York, NY: Elsevier, ISBN*, vol. 444002057, p. 128, 1977.

[4] T. J. McCabe, "A complexity measure. Software Engineering," *Software Engineering, IEEE Transactions on*, vol. 2, no. 4, pp. 308–320, 1976.

[5] J. C. Munson and T. M. Khoshgoftaar, "The detection of fault-prone programs," *Software Engineering, IEEE Transactions on*, vol. 18, no. 5, pp. 423–433, 1992.

[6] V. Y. Shen, T.-J. Yu, S. M. Thebaut, and L. R. Paulsen, "Identifying error-prone softwarean empirical study," *Software Engineering, IEEE Transactions on*, vol. 11, no. 4, pp. 317–324, 1985.

[7] A. Mockus and L. G. Votta, "Identifying reasons for software changes using historic databases," in *Software Maintenance, 2000. Proceedings. International Conference on*, pp. 120–130, IEEE, 2000.

[8] Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross-company software defect prediction," *Information and Software Technology*, vol. 54, no. 3, pp. 248–256, 2012.

[9] J. Nam, S. J. Pan, and S. Kim, "Transfer defect learning," in *Proceedings of the 2013 International Conference on Software Engineering*, pp. 382–391, IEEE Press, 2013.

[10] M. Li, H. Zhang, R. Wu, and Z.-H. Zhou, "Sample-based software defect prediction with active and semi-supervised learning," *Automated Software Engineering*, vol. 19, no. 2, pp. 201–230, 2012.

[11] Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu, "A general software defect-proneness prediction framework," *Software Engineering, IEEE Transactions on*, vol. 37, no. 3, pp. 356–370, 2011.

[12] H. He and E. A. Garcia, "Learning from imbalanced data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, no. 9, pp. 1263–1284, 2009.

[13] R. Pearson, G. Goney, and J. Shwaber, "Imbalanced Clustering for Microarray Time-Series," in *Proc. Intl Conf. Machine Learning*, vol. 3, 2003.

[14] M. Kubat, R. C. Holte, and S. Matwin, "Machine Learning for the Detection of Oil Spills in Satellite Radar Images," *Machine Learning*, vol. 30, no. 2-3, pp. 195–215, 1998.

[15] H. He and X. Shen, "A Ranked Subspace Learning Method for Gene Expression Data Classification," in *Proc. Intl Conf. Artificial Intelligence*, pp. 358–364, 2007.

[16] G. M. Weiss and F. Provost, "Learning when training data are costly: The effect of class distribution on tree induction," *Journal of Artificial Intelligence Research*, vol. 19, pp. 315–354, 2003.

[17] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-Sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, no. 1, pp. 321–357, 2002.

[18] D. Yan, A. Chen, and M. I. Jordan, "Cluster forests," *Computational Statistics & Data Analysis*, vol. 66, pp. 178–192, 2013.

[19] D. F. Specht, "Probabilistic neural networks," *Neural Networks*, vol. 3, no. 1, pp. 109–118, 1990.

[20] L. Behera and M. Nayak, "Comparative study of probabilistic neural network and radial basis functional link neural network in time series data mining," *International Journal of Computer Science and Information Technologie*, vol. 2, no. 6, pp. 2637–2644, 2011.

[21] J. S. Shirabad and T. J. Menzies, "The PROMISE repository of software engineering databases," *School of Information Technology and Engineering, University of Ottawa, Canada*, vol. 24, 2005.

[22] V. Jayaraj and N. S. Raman, "Software Defect Prediction using Boosting Techniques," *International Journal of Computer Applications,(0975–8887)*, vol. 65, no. 13, pp. 1–4, 2013.

[23] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 42, no. 4, pp. 463–484, 2012.

[24] T. Fawcett, "ROC Graphs: Notes and Practical Considerations for Data Mining Researchers," *Technical Report HPL-2003-4*, 2003.

[25] T. Fawcett, "An Introduction to ROC Analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.