Indonesian Journal of Electrical Engineering and Computer Science
Vol. 15, No. 3, September 2019, pp. 1517~1526
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v15.i3.pp1517-1526

Optimized neuro-PSO-based software maintainability prediction using relief features selection method

N. Baskar, C. Chandrasekar
Department of Computer Science, Government Arts College, Udumalpet, TN, India

Article Info

Article history:

Received Jan 1, 2019
Revised Mar 10, 2019
Accepted Apr 25, 2019

Keywords:

Dimensionality reduction
Neuro-PSO
Object oriented
Relief feature
Software maintenance

ABSTRACT

The recent development in software engineering reveals the importance of software maintenance during the time of software development that is becoming more important in software development environment and software metrics, which are very essential for measuring the maintainability of software, software complexity, estimating size, quality and project efforts. There are various approaches through which one can estimate the software cost and predict on various kinds of deliverable items. This paper aims at developing an optimized Neuro-PSO-based software maintainability prediction model by applying the dimensionality reduction using relief feature selection method for identifying the optimal feature subsets in order to increase the accuracy and reduce the time complexity of the prediction model. The simulation result proves the performance of the proposed model which will be more beneficial for the software developers in predicting the maintenance of the software in advance.

Corresponding Author:

N. Baskar,
Research scholar, Department of Computer Science,
Government Arts College,
Udumalpet, TN, India.
Email: n_bas@rediffmail.com

1. INTRODUCTION

Object-Oriented design is more beneficial in software development environment. Object-oriented design metrics is an essential feature to measure software quality over the environment [1]. Object-oriented design is those design which contained all the properties and quality of software that is related to any large or small project [2]. It is a degree through which a system object can hold a particular attribute or characteristics. Object-oriented is a classifying approach that is capable of classifying the problem in terms of object and it may provide many paybacks on reliability, adaptability, reusability and decomposition of problem into easily understandable objects and providing some future modifications [3].

Software metrics makes it possible for a software engineer to measure and predict software as it is necessary resource for a project and projectwork product relevant to the software development effort. Metrics provide the insights that are necessary to create and design model through the test. It also provides a quantitative way to access the quality of internal attributes of the product. Thereby, it enables the software engineer to access the quality before the product is built [4]. Metrics are the crucial source of information through which a software developer takes a decision for designing good software. Some metrics may be transformed to serve their purpose for a new environment.

1.1. Problem Definition

For real world concept-learning problems, feature selection is important to speed up learning and to improve concept quality. This paper reviews and analyzes the past approaches to feature selection and note their strengths and weaknesses. This work introduced and theoretically examined a new algorithm Relief which selects relevant features using a statistical method. Relief does not depend on heuristics. It is accurate even if features interact and is noise-tolerant. It requires only linear time in the number of given features and the number of training instances, regardless of the target concept complexity. The algorithm also has certain limitations such as non-optimal feature set size. The ways to overcome the limitations are suggested. It also reports the test results of comparison between Relief and other feature selection algorithms. The empirical results support the theoretical analysis, suggesting a practical approach to feature selection for problem of reality.

2. RELATED WORK

Khoshgaftaar et al. [5] predicted software quality by using the neural networks as a tool. They classified the modules as either fault-prone or non-fault-prone in a large telecommunications system. They had also made a comparison between the ANN model and a non-parametric discriminate model, in which the ANN model was found to have better predictive accuracy than the other one. Fenton and Neil [6] estimated various software defect prediction models by using size and complexity metrics for predicting defects. They compared fault-proneness estimation models and summarized that software quality is a crucial prerequisite in the system development.

Muthana et al. [7] used the polynomial regression to establish the relationship between design level metrics and the corresponding maintainability of Industrial software. The results have shown that predicted values using polynomial regression were quite close to actual values. Fioravanti and Nesi [8] presented a metric analysis to identify which metrics would be better ranked for its impact on the prediction of adaptive maintenance for object-oriented systems. The model and metrics proposed have been validated against real data by using MLR (Multi-linear Regression Analysis) Model. The validation has identified that the several metrics can be profitably employed for the prediction of software maintainability.

Dagpinaret al. [9] also based their study on empirical data to establish the relationship between software metrics and its maintainability. However instead of designing level metrics of structure languages, the metrics were replaced by object-oriented metrics. They recorded significant impact of two metrics i.e. direct coupling metric and size metric on software maintainability while other parameters like cohesion, inheritance and indirect coupling were not considered significant by them. Thwin and Quah [10] used neural networks to build software quality prediction models. They proposed that maintainability could be estimated with the help of fuzzy model. They also proved empirically that the integrated measure of maintenance obtained from this fuzzy model has strong correlation with maintenance.

Zhou and Leung [11] have used multivariate adaptive regression splines (MARS) for predicting object-oriented software maintainability in 2007. They compared the prediction accuracy of the proposed model with four other prevailing models: multivariate linear regression (MLR), support vector regression (SVR), artificial neural network(ANN), and regression tree (RT) and stated that MARS is the best model to be used as far as maintainability of prediction is concerned. Hu and Zhong [12] proposed a model based on neuralnetwork to predict software module risk. The learning vector quantization network used in their study has predicted software quality. Arvindar et al. [13] predicted the software maintenance effort by application of diverse soft computing techniques. Two commercial software products were taken as dataset and they observed that soft computing techniques are useful for the construction of accurate models to speculate the maintenance effort. In their analysis, maintenance effort was chosen as dependent variable and eight Object-Oriented metrics as independent variable.

Ratra et al. [14] compared early prediction of fault-prone modules in software design and for this they have applied clustering and neural network techniques. The performance of the two methods were measured based on their accuracy, the mean absolute error, and root mean square error values. Their result signified that the performance of neural network approach is much superior to clustering based approach. Malhotra, Chug [15] aimed at assessing the efficiency of different prediction models for prediction maintainability of web-based systems using Object- Oriented metrics. Ping [16] used Hidden Markov Model (HMM) to define health index of a product in literature and suggested that it works as a weight on the process of maintenance behavior over a period of time.

Baskar and Ramani [17] in their work presented measurements of Coupling between Object (CBO) in object-oriented programming. The metric values of class and interface inheritance diagrams have been compared to prove whether maintainability is improved to use and beneficial for the software developers. Baskar et al. extended their work in [18] by devising new metric for software maintainability using Cognitive Weighted Response for a Class (CWRFC). The proposed metric is applied to the computer classification and acquired better results which will help not only for low maintenance of the component-based system but also to reduce the complexity efforts.

2.1. Dataset Description

In this paper, the maintenance effort data are obtained from Object-Oriented software data sets namely User Interface System (UIMS) and Quality Evaluation System (QUES) for computing the maintenance effort. The software's system, UIMS have 39 and QUES have 69 classes. The maintainability of software is measured by the number of lines changed per class. The attributes used are Depth of Inheritance Tree (DIT), Weighted Method Complexity (WMC), Number of Children (NOC), Coupling between Objects (CBO), Lack of Cohesion of Methods (LCOM), Messaging Passing Coupling (MPC), Response for a Class (RFC), Data Abstraction Coupling (DAC), Number of Methods (NOM), Size 1, Size 2, Change.

3. PROPOSED METHOD

The proposed method enhances the outcome of the software maintenance on object-oriented software dataset to determine the effort of maintenance. The dataset consists of the metrics of the two softwares UIMS and QUES. The total number of attributes in the original dataset is 12 but all these features are not necessary for maintainability analysis. Hence, the important features are determined using the relief feature subset algorithm by ranking each feature with its computed weight and our previous works [17, 18] prove the importance of Coupling Between Objects(CBO), Depth of Inheritance Tree (DIT), Cognitive Weighted Response for a Class (CWRFC) are significantly considered for finding the severity of maintenance in both classes and interfaces. The potential feature set used for predicting the severity of the maintainability of each classes are as follows:

Depth of Inheritance Tree (DIT), Weighted Method Complexity (WMC), Cognitive Weighted Response for a Class (CWRFC), Data Abstraction Coupling (DAC), Number of Methods (NOM), Lack of Cohesion of Methods (LCOM) and Messaging Passing Coupling (MPC). These are now fed as the input to the Particle Swarm Optimization-based artificial neural network for predicting and classifying them based on the class variable change. The change metric is used as the prediction variable for this model. This model consists of three layers with seven input nodes in the input layer, 5 nodes as hidden nodes in the hidden layer, 1 output node in the output layer.

The model is trained using the Back Propagation-based neural network whose initial weights of the nodes are assigned randomly. The predicted output is compared with the expected output. If there is a variance among them, the weights are reassigned with the help of the particle swarm optimization to each hidden nodes in an optimal way and the iteration is continued till the assigned criteria is met. Figure 1 shows optimised neuro-PSO using relief feature selection algorithm

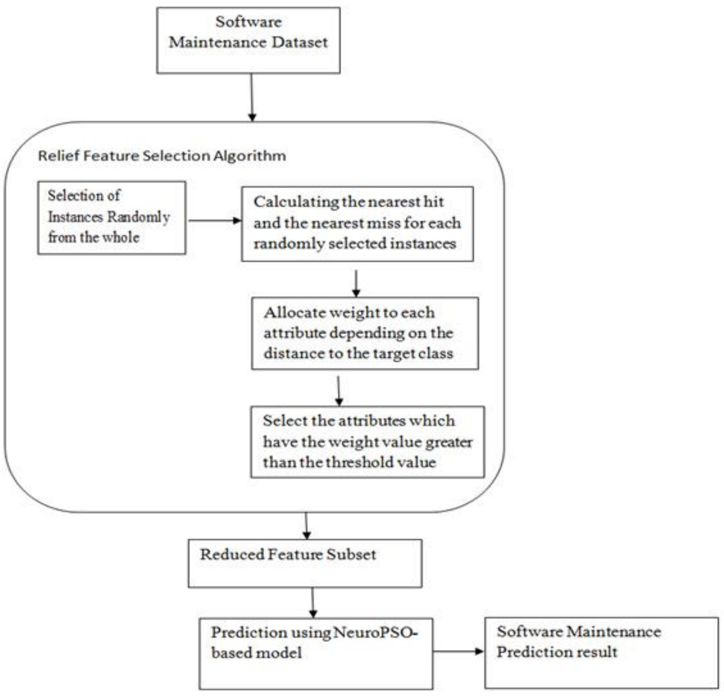
Software Maintenance Dataset
Relief Feature Selection Algorithm
Selection of Instances Randomly from the whole
Calculating the nearest hit and the nearest miss for each randomly selected instances
Allocate weight to each attribute depending on the distance to the target class
Select the attributes which have the weight value greater than the threshold value
Reduced Feature Subset
Prediction using NeuroPSO-based model
Software Maintenance Prediction result

Figure 1. Optimised Neuro-PSO using relief feature selection algorithm

3.1. Relief Algorithm

Relief algorithm is able to detect the conditional dependencies between attributes for feature selection. Relief algorithm is considered to be a feature subset selection algorithm at the time of pre-processing the dataset [19]. Relief algorithm is viewed as one of the successful pre-processing algorithms which are assumed as two-class classification problems. An instance is represented by a vector composed of p feature values. ‘S’ denotes a set of training instances with size ‘n’. ‘F’ is the given feature set {f1,f2,..,fp}. An instance ‘X’ is denoted by a ‘p-dimensional vector’ (x1,x2,..xp), where ‘xj’ denotes the value of feature ‘fj’ of ‘X’. Relief is a feature selection algorithm inspired by instance-based learning [19].

Given training data ‘S’, sample size ‘m’, and a threshold of relevancy ‘τ’ encodes a relevance threshold ($0 \leq 2; t \leq 1$). It is assumed that the scale of every feature is either nominal (including Boolean) or numerical (integer or real). Differences of feature values between two instances ‘X’ and ‘Y’ are defined by the following function diff.

When x_k and y_k are nominal,

$$\text{diff}(x_k,y_k) = \begin{cases} 0 & x_k = y_k \\ 1 & x_k \neq y_k \end{cases}$$

When x_k and y_k are numerical,

$\text{diff}(x_k,y_k)= (x_k- y_k)/nu_k$

where nu_k is a normalization unit to normalize the values of difference into the interval [0,1]

The complexity of Relief is Θ (pmn) because the distance between ‘X’ and each of the ‘n’ instances is calculated, taking Θ (p) time, to determine its Near-miss and Near-hit inside a loop iterating ‘m’ times. ‘m’ is a constant value affecting the accuracy of relevance levels. Since ‘m’ is chosen independently of ‘p’ and ‘n’, the complexity is Θ (pn). Thus, the algorithm can select statistically relevant features in linear time in terms of the number of feature and the number of training instances.

```
Relief(δ,m,τ)
    Separate δ into $δ^+$ = {positive instances} and
$δ^-$ = {negative instances}
   W = (0,0,..,0)
   For i = 1 to m
           Pick at random an instance $X \in S$
            Pick at random one of the positive instances
closest to X,$Z^+ \in δ^+$
Pick at random one of negative instances
closest to X,$Z^- \in δ^-$
if (X is a positive instances)
then Near-hit = $Z^+$ ; Near-miss = $Z^-$
else Near-hit = $Z^-$ ; Near-miss = $Z^+$
update-weight(W,X,Near-hit,Near-miss)
Relavance = (1/m)W
For i=1 to p
if (relavance$_i \geq τ$
then $f_i$ is a relevant feature
else $f_i$ is a irrelevant feature
update-weight(W,X,Near-hit,Near-miss)
       For i=1 to p
$W_i = W_i - \text{diff}(x_i,\text{near-hit}_i)^2 + \text{diff}(x_i,\text{near-miss}_i)^2$
```

3.2. Dimensionality Reduced Optimized Features Using Relief Feature Selection Method

In the previous work [20] eleven different metrics were selected for measuring the maintenance severity but this work enhances it by selecting only the optimal metrics which contribute more in the focus of the maintenance of the software precisely. The ranker search method is used in this proposal for ranking the metrics based on their performance in the object-oriented software maintenance. The metrics with highest ranking value are given more importance and some of the metrics are eliminated for their high dependency. The reduced metrics are discussed in detailed in the following sections.

3.2.1 Depth Of Inheritance Tree (DIT)

The DIT metric measures the position in the inheritance hierarchy [11]. The DIT metric addresses the inheritance concept. It is logical that the lower class in the inheritance tree, can access more super-class properties owing due to its inheritance. If the sub-class inherits properties from the super-class without using the methods defined in the superclass, the encapsulation of the super-class is violated. One may hypothesize that the larger the DIT metric, the harder it is to

DIT = inheritance level number; ranging from 0 to N; where N is a positive integer.

Maintain the class. The calculation of the DIT metric is the level number for a class in the inheritance hierarchy. The root class' DIT is zero.

3.2.2 Response For Class (RFC)

The RFC metric measures the cardinality of the response set of a class. The response set of a class consists of all local methods and all the methods called by local methods [11]. It seems logical that the larger the response set for a class, the more complex the class. One may intuit that the larger the RFC metric, the harder it is to maintain the class, since calling a large number of methods in response of a message makes tracing an error difficult. The calculation of RFC [18] is:

RFC = Number of elements in RS
Where RS is the response set for the class. It can be expressed as

RS=Union of methods in the class and inherited methods from the parent class
$RS = M \cup IM$

Where IM = set of inherited methods, M = Set of methods in the class

3.2.3 Weighted Method Complexity (WMC)

The WMC metric means the static complexity of all the methods [11]. This metric addresses the class and method concepts. It is logical that the more the methods, the more complex the class. If there are more control flows in a class methods, it will be harder to understand and maintain them. The WMC is calculated as the sum of McCabe's cyclomatic complexity of each local method:

WMC = summation of the McCabe's cyclomatic complexity of all local methods;
Ranging from 0 to N; where N is a positive integer.

3.2.4 Message Passing Coupling (MPC)

The Message Passing Coupling (MPC) is used to measure the complexity of message passing among classes. since the pattern of the message is defined by a class, it is used by objects of the class. The MPC metric also gives an indication of how many messages are passed among objects of the classes:

MPC = number of send-statements defined in a class.

The number of messages send out from a class may indicate how dependent the implementations of the local methods are upon the methods in other classes. This may not be indicative of the number of messages received by the class.

3.2.5 Number of Methods (NOM)

Another metric used in this research is the Number of Methods (NOM) in a class. Since the local methods in a class constitute the interface increment of a class, NOM serves the best as an interface metric. NOM is easy to collect in most object-oriented programming languages.

NOM = number of local methods;

3.2.6 Cognitive Weighted Response For A Class (CWRFC)

The CWRFC is used to calculate the maintainability of the class using the Response Set Complexity (RSC). If there are 'm' numbers of response sets in a class, then the CWRFC of that class can be calculated by:

$$\mathrm{CWRFC} = \sum_{j=1}^{m} \mathrm{RSC}_j$$

Where, RSC is the Response Set Complexity, which can be calculated by adding the set of all methods (M) in a class and set of methods (R)

3.2.7 Proposed Algorithm Of Optimized Neuro-Pso Software Maintenance-Based Prediction Model

Input: QUES and UIMS Dataset

Steps

1. Collect the dataset from the two different system softwares
2. Perform feature selection using Relief Feature Selection Algorithm
   i) Select Instance Randomly from the given dataset
   ii) Calculate the nearest hits and nearest miss with other instances
   iii) Assign weight to each features based on the distance towards the target class
   iv) Find the mean value of the instances and its weight
   v) Select the features which have the weight value greater than the threshold value which is assigned by the relief feature selection method
   vi) The reduced features are considered as potential features for determining the software maintenance among the interfaces and classes
3. The selected potential attributes are used as the input for Neuro-PSO classifier
4. Construct the model using input, hidden and output layer
5. Train the model using back propagation based artificial neural network
6. The weights are reassigned using the particle swarm optimization using its parameter's velocity and position
7. Test the model for predicting the severity of the software maintenance effectively using the proposed model and
8. Iterate the process till the goal is met

Output: Reduced feature set, Classification

4. EXPERIMENTAL RESULT

This proposed Optimized Neuro-PSO-based software maintenance prediction model is designed and simulated using MATLAB. The dataset is collected from two different Object-Oriented software data sets User Interface System (UIMS) andQuality Evaluation System (QUES). This proposed work focuses on optimizing the performance of the neuro-pso-based software maintenance prediction by adapting the dimensionality reduction approach using relief feature selection algorithm. The reduced feature set is then used as the input for the neuro-pso model for predicting the severity of the maintenance of each class of the softwares. The proposed work is compared with existing approaches and the simulation result shows the promising output of the proposed work compared to the existing approaches namely ANN and Neuro-PSO.

4.1. Evaluation Metrics

This subsection describes the various metrics used for determining the performance of the proposed work with the existing approaches namely GMDH, GRNN, PNN.

The Precision is a measure of percentage of positively predicted instances which are actual positive and they are calculated using the formula

$$\Pr ecision = \frac{\text{Truly classified positive instances}}{\text{Truly classified positive instances} + \text{Falsely classified as negative instances}}$$

The measure Recall is a percentage of real positive instances which are predicted positive instances and the formula is

$$\mathrm{Re}\,call = \frac{\text{Truly classified as positive instances}}{\text{Truly classified as positive instances} + \text{Falsely classified as positive instance}}$$

The F-measure is the combination mean of both precision and recall

$$F-Measure = \frac{2 * \text{Truly classified positive instances}}{2 * \text{Truly classified positive} + \text{Falsely classified positive} + \text{Falsely classified negative}}$$

Magnitude of Relative Error (MRE): It is measured by taking the absolute value of the difference between the actual value and the predicted value. The formula for this measure is:

$$\text{MRE} = \frac{|Actualvalue - predictedvalue|}{Actualvalue}$$

Mean Magnitude of Relative Error (MMRE): MMRE is the mean of MRE. The formula for this measure is:

$$\text{MMRE} = \sum_{I=1}^{N} MRE_I$$

Pred: Pred is measured by the predicted values whose MRE is less than or equal to a specified value. ‘k’ is the number of predicted values which are less than or equal to the specified value. ‘q’ is the specified value and ‘N’ is the total number of cases.

$$\text{Pred(q)} = \frac{K}{N}$$

4.2. The Network Topology of Neuron Networks

In this study, the BP neuron network has three layers including one hidden-layer. The neural networks models are trained with 7 neurons as input data while 5 neurons for the hidden layer, and 1 neuron for output layer. The neuron transferring function in hidden-layer is sigmoid function. In matlab represented as tansig. In output-layer purely linear is represented as purelin. And the training function is traingdm. The training error precision is 0.0001.

4.3. Parameters of PSO

The parameters of PSO were selected as follows. The initial location and velocity of search point is randomly generated between [-1, 1]; the maximum velocity of particles is 0.5; the population size is 40; the maximum times of iteration is 30000; the accelerated coefficients c1 =2.3, c2=1.8; the inertia weight is gradually decreased from 0.90 to 0.40 in order to reduce the influence of past velocity, and the particle dimension is 19. The Table 1 shows the outcome of the relief feature selection method. The whole dataset consists of twelve features with one class attribute. The Relief feature selection method determines the weight of each feature based on the contribution of them towards the target feature by determining the instances misses and the hits. Its mean value is determined and assigned as the final weight value of each class. The features are ranked according to their obtained weights. This feature selection algorithm selects 7 features as potential features and they are ranked according to their obtained weight value. The table depicts that the wmc holds the highest rank value of 0.0624, dit obtains 0,0370, dac obtains 0.0360, mpc obtains 0.0227, the cwrfc, nom and lcom holds the ranking value of 0.0197, 0.0182 and 0.0157 respectively. Thus the reduced dimensionality consists of only seven features.

Table 1. Dimensionality Reduction using Relief Feature Selection Method

Ranking Value	Attribute No	Attributes Name
0.0624	6	wmc
0.0370	1	dit
0.0360	5	dac
0.0227	2	mpc
0.0197	3	cwrfc
0.0182	7	nom
0.0157	4	lcom

The Table 2 and the Figure 2 depict the performance of the three different sets of features and the performance of the ANN classifier. The simulation result shows that, due to the high dependence among the features, the whole feature set and the ID3-based feature set produces less performance compared to the proposed relief feature selection method. This proposed work consists of high precision value of 0.892, Recall value as 0.854 and the F-measure value as 0.876. The Table 3 and the F igure 3 depict the performance of the three different sets of features and the performance of the Neruo-PSO-based classifier model. The proposed Neuro-PSO optimized its result by eliminating irrelevant and redundant features from the whole features set, so that it achieves higher accuracy in the terms of Precision as 0.932, Recall as 0.964 and F-measure as 0.989. It is also observed that the overall performance of the neuro-pso is considerably increased in accuracy while comparing the artificial neural network because of the heuristic knowledge in assigning the weights to the nodes using the particle swarm optimization.

Table 2. Performance of the Artificial Neural Network-based Classifier

	Precision	Recall	F-Measure
Whole Features	0.721	0.725	0.727
Relief Feature Selection	0.892	0.854	0.876
ID3 Feature Selection	0.85	0.837	0.865

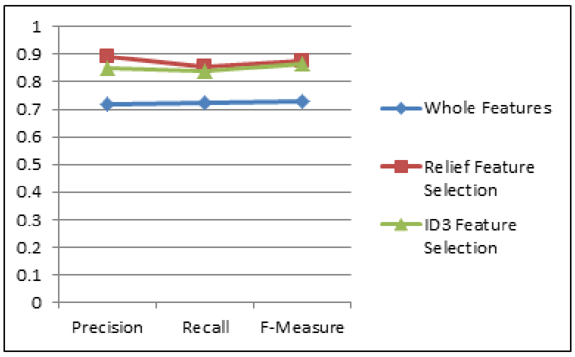

Figure 2. Performance Result of the Artificial Neural Network-based Classifier

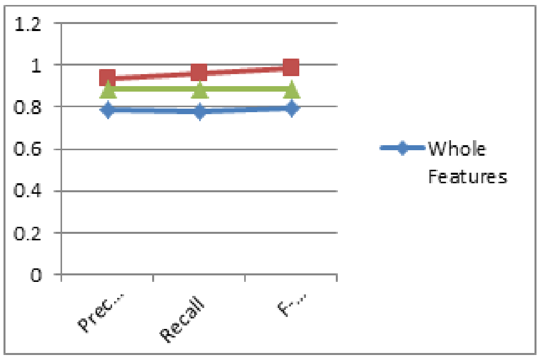

Figure 3. Performance Result of the Neuro-PSO-based Classifier

Table 3. Performance of the Neuro-PSO-based Classifier

	Precision	Recall	F-Measure
Whole Features	0.791	0.782	0.794
Relief Feature Selection	0.932	0.964	0.989
ID3 Feature Selection	0.886	0.889	0.887

From the Table 4 and Figure 4 it is observed that the time taken for performing the software maintenance prediction is highly reduced while using the optimized Neuro-PSO-based classifier model, because they performed dimensionality reduction to determine the potential features as input to the neuro-pso model for predicting and classification during both the training and testing time. The other two methods takes more time in computation because of the size of the feature set used and lack in assignment of the weights to the nodes during computation. The optimized neruo-pso takes 0.52 sec for training the dataset and 0.48 sec for testing the dataset. The Table 5 and the Figure 5 show the Error Ratio and the prediction accuracy of the proposed optimized Neuro-PSO with the dimensionality reduction using relief feature selection method. The other existing approaches holds the high error rate and low prediction value because of considering the high depended and correlated features for classification. The proposed method contains the max MRE value of 1.95387 and MMRE value as 0.2262 because of considering only the relevant attributes for feature selection and prediction value for 0.25 is 0.3803 and for 0.75 is 0.6934 which are relatively high comparing to the existing methods because of its optimized performance.

Table 4. Performance Comparison of Optimized Neuro-PSO with existing approaches based on Time Complexity

Method	Training Time (sec)	Testing Time (sec)
Optimized Neuro-PSO Classifier	0.52	0.48
Neuro-PSO classifier	0.65	0.43
Artificial Neural Network	0.72	0.35

Table 5. Performance comparison of the proposed optimized Neuro-PSO with the existing approaches

	Measures			
Models Used	Max MRE	MMRE	Pred(0.25)	Pred(0.75)
GMDH	3.42656	0.3341	0.2894	0.5263
GRNN	2.40739	0.3094	0.2987	0.5526
PNN	3.05611	0.3353	0.2631	0.5526
NPSO	2.02547	0.2931	0.2998	0.5612
Optimized NPSO	1.95387	0.2262	0.3803	0.6934

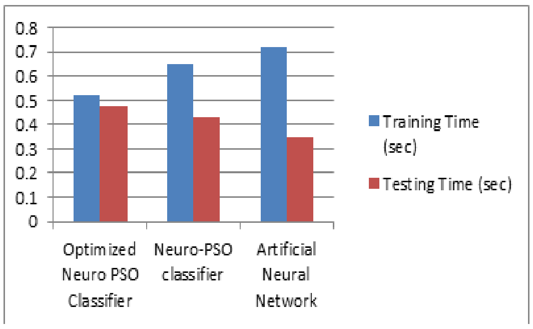

Figure 4. Performance Comparison Result of Optimized Neuro-PSO with existing approaches based on Time Complexity

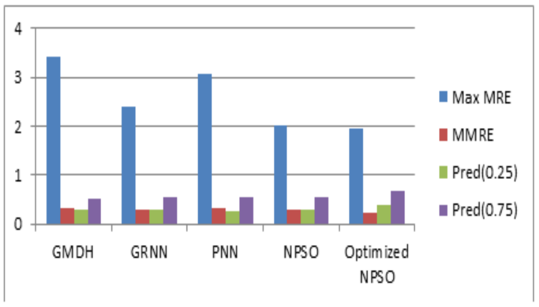

Figure 5. Performance comparison Result of the proposed optimized Neuro-PSO with the existing approaches

5. CONCLUSION

The major objective of this proposed work is to optimize the performance of the software maintenance prediction. This is achieved using the two different phases they are dimensionality reduction using relief feature selection and increasing the prediction accuracy using the Neuro-PSO. The object-oriented softwares are mainly relied on increased understanding of the state of the software metrics. The increasing complexity in software maintenance highly degrades the quality of the software due to frequent modification in the functioning of the software. This proposed work achieves optimized result in developing a well-equipped software maintenance prediction model for object-oriented softwares.

REFERENCES

[1] Neelamegam C, Punithavali M. A survey on object oriented quality metrics, *Global journal of computer science and technologies*, 2011, pp 183-186.
[2] Deepak A, Pooja K, Alpika T, Sharma S. Software quality estimation through object oriented design metrics, IJCSNS International journal of computer science and network security, April 2011, pp 100-104.
[3] Henderson A, Seller. Object oriented metrices: measure of complexity, Prentice Hall, 1996.
[4] Shaik A, Reddy C.P.K, Manda B, prakashine, Deepti K. Metrics for object oriented design software system: A Survey, *Journal of emerging trend in engineer and applied science (JETEAS)*, pp 190- 198, 2010
[5] Khoshgaftaar T.M, Allen E.D, Hudepohl J.P and Aud S.J. Application of neural networks to software quality modelling of a very large telecommunications system, *IEEE Transactions on Neural Networks*, Vol. 8, No. 4, 1997, pp. 902--909.
[6] Fenton N. E and Neil M. A Critique of Software Defect Prediction Models, *IEEE Trans. Software Engineering*, 1999, vol. 25, Issue no. 5,pp. 675-689.
[7] Muthanna S, Kontogiannis K, Ponnambalam K, Stacey B. *A Maintainability Model for Industrial Software Systems Using Design Level Metrics*, Proceeding of Seventh Conference on Reverse Engineering, IEEE Computer Society, pp. 248, 2000.

[8] Fioravanti F and Nesi P. Estimation and prediction metrics for adaptive maintenance effort of object oriented systems, *IEEE Transactions on Software Engineering*, vol. 27, no. 12, pp. 1062-1084, 2001.

[9] Dagpinar M. and Jahnke J.H. *Predicting Maintainability with Object-Oriented Metrics-An Empirical Comparison*, Proceedings of the 10th Working Conference on Reverse Engineering, pp 155-164, Nov 2003.

[10] Thwin M and Quah T. Application of neural networks for software quality prediction using object oriented metrics, *Journal of Systems and Software*, vol. 76, no. 2, pp. 147-156, 2005.

[11] Zhou Y, Leung H (2007) Predicting object-oriented software maintainability using multivariate adaptive regression splines. JSystSoftw 80(8):1349–1361. doi:10.1016/j.jss.2006.10.049.

[12] Hu Q and Zhong C. Model of predicting software module risk based on neural network, *Computer Engineering and Applications*, Vol.43, No.18, pp.106-110, 2007.

[13] Arvindar Kaur, Kaur K and Malhotra R. Soft Computing Approaches for Prediction of Software Maintenance Effort, *International Journal of Computer Applications*, Vol. 1, no.16, 2010.

[14] Ratra R, Randhawa N.S, Kaur P, Singh G. Early Prediction of Fault Prone Modules using Clustering Based vs. Neural Network Approach in Software Systems, *IJECT* Vol. 2, Issue 4, Oct .–Dec.2011.

[15] RuchikaMalhotra, Anuradha Chug. Application of Group Method of Data Handling model for software maintainability prediction using object oriented systems, *Springer* Int J SystAssurEngManag DOI 10.1007/s13198-014-0227-4,2014.

[16] Ping L. A Quantitative Approach to Software Maintainability Prediction, *International Forum on Information Technology and Applications*, Vol: 1, No : 1, pp : 105-108, July 2015.

[17] Baskar N, Ramani A.V. Determining Software Maintainability Of Java Interfaces Using Quantitative Approach, *Journal of Theoretical and Applied Information Technology (JATIT)*, ISSN:1992-8645, E-ISSN: 1817-3195, Vol. 69 No.2, Pg:318-325, 20th November 2014.

[18] Baskar N, Ramani A.V, Chandrasekar C. An Interface Maintainability Measure For Component-Based Software Systems, *Asian Journal of Information Technology (AJIT)*, ISSN:1682-3915,Vol:16, No:6, Pg:503-510, 2017.

[19] Durgabai P.L. Feature Selection Using Relief Algorithm, *International Journal of Advanced Research in Computer and Communication Engineering,* Vol. 3, Issue 10, October 2014.

[20] Baskar N, Chandrasekar C. An Evolving Neuro-PSO-based Software Maintainability Prediction, *International Journal of Computer Applications*, ISSN: 0975-8887, Vol:179, No:18, Pg:7-14, February 2018.