Early Stage Software Effort Estimation using Random Forest Technique based on Optimized Class Point Approach

SHASHANK MOULI SATAPATHY[1]
BARADA PRASANNA ACHARYA[2]
SANTANU KUMAR RATH[3]

Department of Computer Science and Engineering
National Institute of Technology, Rourkela
Rourkela - 769008, Odisha, India
[1]shashankamouli@gmail.com
[2]barada.p.acharya@gmail.com
[3]skrath@nitrkl.ac.in

Abstract. Evaluating software development effort remains a complex issue drawing in extensive research consideration. The success of software development depends very much on proper estimation of effort required to develop the software. Hence, correctly assessing the effort needed to develop a software product is a major concern in software industries. Random Forest (RF) technique is prevalently utilized machine learning techniques that aides in getting enhanced evaluated values. The main research work carried out in this paper is to accurately estimate the effort required in developing various software projects by using the optimized class point approach (CPA). Then, optimization of the effort parameters is achieved using the RF technique to obtain better prediction accuracy. Furthermore, performance comparisons of the models obtained using the RF technique with other machine learning techniques such as the Multi-Layer Perceptron (MLP), Radial Basis Function Network (RBFN), Support Vector Regression (SVR) and Stochastic Gradient Boosting (SGB) techniques are presented in order to highlight the performance achieved by each technique.

Keywords: Class Point Approach, Object Oriented Analysis and Design, Random Forest, Software Effort Estimation.

(Received September 11th, 2014 / Accepted January 20th, 2015)

1 Introduction

Object-oriented (OO) technology is the accepted methodology for software development in major industries. Various features are offered by object oriented programming concepts, which play an important role during software development process [3]. With the increase in the complexities associated with modern day software projects, the need for early and accurate effort estimation in the software development phase has become pivotal. Currently used effort estimation techniques like Function Point (FP) and COCOMO, fail to consistently estimate the cost and effort required to develop the software [20]. These techniques are not capable of measuring cost and effort because they are tailored for procedural-oriented software development paradigm [12, 21]. The procedural oriented paradigm and object-oriented paradigm differ because the former splits the data and procedure; while the latter combines them.

As far as effort estimation is concerned, a number of unsolved problems and errors still exist. Accurate estimation of a software project is always important for determining the feasibility of the project [19]. In the present scenario, most software project planning depends upon accurate effort estimation. Since *class* is the fundamental logical unit of an OO system, the utilization of the class point methodology to compute the project effort serves to get a improved result. During the calculation procedure of the final adjusted class point, two measures, CP1 and CP2, are utilized. CP1 is figured utilizing two measures, Number of External Methods (NEM) and Number of Services Requested (NSR); whereas CP2 is ascertained by utilizing an alternate metric as a part of expansion to NEM and NSR, Number of Attributes (NOA). NEM figures the measure of the interface of a class and is directed by the measure of local public methods, although NSR gives a measure of the linkage of the components of the software system. On the other hand, NOA helps in finding out the number of attributes utilized in a class. In case of function point approach (FPA) and CPA, the Technical Complexity Factor (TCF) is calculated based on the impact of various general characteristics of a system. However, in both these cases, the non-technical factors such as effectiveness of the management, technical competence of developers, security of the system, system's reliability, system's maintenance capability and system's portability are not looked into [25]. Hence in this study, the optimized CPA is utilized to ascertain the effort needed to create the software adopting these six non-technical factors. The fuzzy logic approach is used to optimize the complexity value of various types of class which in-turn optimizes the final estimated class point value. Likewise with a specific end goal to accomplish better prediction accuracy, a Random Forest (RF)-based effort estimation model is applied over the obtained optimized class point value. The results obtained from the RF-based estimation model are then compared with the results obtained from other machine learning techniques i.e., MLP, RBFN, SVR and SGB-based models in order to access their performance. Results proved that the RF technique-based software effort estimation model outperforms other models.

2 Related Work

Costagliola et al. [5] observed that the prediction accuracy of CP1 and CP2 under the class point approach were 75% and 83% respectively. They drew this conclusion by conducting an experiment on a dataset with forty projects. Zhou and Liu [25] extended this methodology by including an alternate measure CP3 and considered twenty four attributes rather than the eighteen acknowledged by Gennaro Costagliola et al. By utilizing this methodology, they watched that the performance of CP1 and CP2 stay unaltered, although the number of characteristics changed. Kanmani et al. [9] utilized the same CPA with the ANN model for mapping CP1 and CP2 into the assessed software development effort and observed that the prediction accuracy for CP1 was enhanced to 83% and CP2 to 87%. Kim et al. [11] presented some new meanings of class point to interpret system's architectural complexity in an improved way. They utilized various additional parameters along with NEM, NSR and NOA to compute the total number of adjusted class point value. Kanmani et al. [10] introduced a novel technique to utilize the CPA with fuzzy logic by embracing the subtractive clustering technique for computing effort and contrasted it with the result acquired from the ANN. They observed that the fuzzy system focused around the subtractive clustering technique outperforms ANN. Sheta [23] used Takagi-Sugeno-Kang (TSK) fuzzy model to develop fuzzy models for two different type of nonlinear processes. The first one is NASA software projects effort estimation process and the second one is the stock market prediction process for S& P 500.

Satapathy et al. [22] proposed a novel SVR based effort estimation technique-based class point approach and obtain promising results. Satapathy et al. [19] also proposed a novel SGB technique-based effort estimation model using class point approach. From the analysis, it is observed that the SGB technique-based effort estimation model provides improved prediction accuracy than MLP and RBFN technique. Elish [6] used multiple additive regression trees as a novel advanced data mining technique that broadens and enhances the classification and regression trees (CART) model using a treeboost technique. The predicted results obtained were then compared with linear regression, RBFN, and SVR models with the help of a NASA software project data set and found an improved estimation accuracy. Nassif et al. [17] presented a novel regression model for software effort estimation focused around use case diagrams. By analyzing the outcome, they proved that the the software effort estimation accuracy can be improved by 16.5% using PRED(25) and 25% using PRED(35). Elyassami et al. [7] investigate the utilization of Fuzzy choice tree for software effort estimation. The proposed model empower to handle questionable and loose information, which enhance the correctness of obtained evaluations.

Nassif et al. [15] proposed a new regression model for software effort estimation based on use case point. Further, they applied a Sugeno fuzzy inference system technique on this model and obtain an improvement of 11% in MMRE value. Nassif et al. [16] also proposed an ANN model to anticipate effort required to develop software from use case diagrams focused around the

UCP model with the assistance of 240 data and obtained an improved result than other regression models. Pahariya et al. [18] proposed a new concurrent architecture for a genetic programming based feature selection algorithm for software effort estimation and compared the predicted effort with other computational intelligence techniques. The results show that the new recurrent architecture design for GA performs better than the other models. Nassif et al. [14] also proposed some other techniques using fuzzy logic and ANN to enhance the correctness of the UCP model and achieved up to 22% improvement in prediction accuracy result.. Huang et al. [8] proposed a neuro-fuzzy technique for software effort estimation and obtained promising results. Baskeles et al. [1] proposed a model that uses machine learning techniques and assess the model using the data collected from public data sets and the data collected from software industries. From investigation, it is discovered that the utilization of any one model can't create the best comes about for software effort estimation.

3 Methodologies Used

The following methodologies are used in this paper to calculate the effort of a software product.

3.1 Class Point Approach (CPA)

The CPA was presented by Gennaro Costagliola et al. in 1998 [4]. It was focused around the FPA methodology to speak to the interior qualities of a software. The essential thought of the CPA system is calculation of classes in a project. It is determined from the perception that in the procedural model functions or methods are the essential programming units; while, in the OO model, classes are the coherent building pieces.

The block diagram, demonstrated in figure 1, states the steps to compute the project development effort using class point approach.

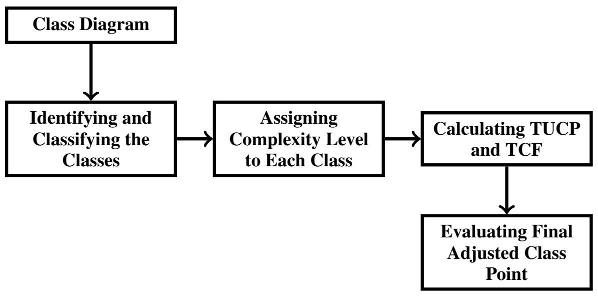
Class Diagram
Identifying and Classifying the Classes
Assigning Complexity Level to Each Class
Calculating TUCP and TCF
Evaluating Final Adjusted Class Point

Figure 1: Final Adjusted Class Point Calculation Steps

The system to acquire the amount of class points is isolated into three principle stages [5].

- Estimating information processing size
  - Identifying and classifying the classes
  - Assigning complexity level for each classified class
  - Calculating the Total Unadjusted Class Points (TUCP) value
- Estimating the Technical Complexity Factor (TCF) value
- Calculating the final value of Adjusted Class Point (CP)

In this study, the above phases are followed to calculate final optimized class points. Detailed descriptions of all the phases were already provided by Gennaro Costagliola et al. [5]. Herein, the total number of optimized class point value is then used as an input parameter to the random forest model to calculate the estimated effort.

3.1.1 Fuzzy Logic System

Fuzzy sets were presented by L. A. Zadeh (1965) as a method for speaking to and controlling information that was not exact, but instead fuzzy. Fuzzy logic gives a derivation morphology that empowers rough human thinking capacities to be connected to information based frameworks [21]. Fuzzy system comprises of three principle segments: *fuzzification process*, *derivation from fuzzy rules* and *defuzzification process*. Among different fuzzy models, the model presented by Takagi, Sugeno and Kang (TSK fuzzy) [24] is more applicable for sample data based fuzzy modeling, on the grounds that it requires less rules. Each rule's outcome with linear function can portray the information yield mapping in a vast reach, and the fuzzy implications utilized within the model is likewise basic. In this study, fuzzy modeling has been utilized to improve the complexity of TUCP and fuzzy subtractive clustering to estimate the effort.

3.2 Random Forest Technique

Random forest (RF) is an troupe learning technique used for classification and regression purposes [2]. It builds a number of decision trees during training period and chooses the final class by selecting the mode of the classes generated by distinctive trees. To obtain better results than the results from individual decision tree models, ensemble model combines the results from different models of similar type or different types.

The concept behind it is that random forests grow many classification trees. To generate many classification trees, a random vector λ and an input vector x is used. A random vector λ_k is produced for the kth tree, which is autonomous of the previous random vectors $\lambda_1, ..., \lambda_{k-1}$, however with the equal distribution. A tree is developed utilizing the training set and λ_k, which generates a classifier $h(x, \lambda_k)$ where x is an input vector. To categorize new object from an input vector, the input vector x is put down each of the trees in the forest. Each tree provides a classification by voting for that class. Then, the classification having the maximum number of votes among over all the trees in the forest is chosen. In case of regression, the prediction accuracy of the forest is obtained by taking the average of the individual tree predictions.

RF for regression purpose are created by developing trees relying upon a random vector λ specified that the tree predictor $h(x, \lambda)$ undertakes numerical data instead of class labels. The output produced by the predictor is $h(x)$ and the actual effort value is Y. For any numerical predictor $h(x)$, the generalized mean-squared error is calculated as

$$E_{x,Y}(Y - h(x))^2 \tag{1}$$

By calculating the average value obtained over k trees $h(x, \lambda_k)$, the RF predictor is modeled.

4 Proposed Approach

The proposed approach is applied over forty data set used in [5]. The utilization of such data set proposes to assess software development effort and provides introductory test evidence of the viability of the CPA. The utilization of this data set helps to evaluate the effort required to develop software and validate the practicability of improvement. In the data set, every row displays the details of one project developed in JAVA language, indicating values of NEM, NSR and NOA for that project. Apart from that, it also displays values of CP1, CP2 and the actual effort (denoted by EFH) expressed in terms of person-hours required to successfully complete the project.These data are utilized to develop the random forest technique based software effort estimation model.

The diagram demonstrated in figure 2 displays the proposed steps used to determine the predicted effort using the random forest technique-based on optimized class point approach.

To compute the software development effort, essentially the accompanying steps are utilized.

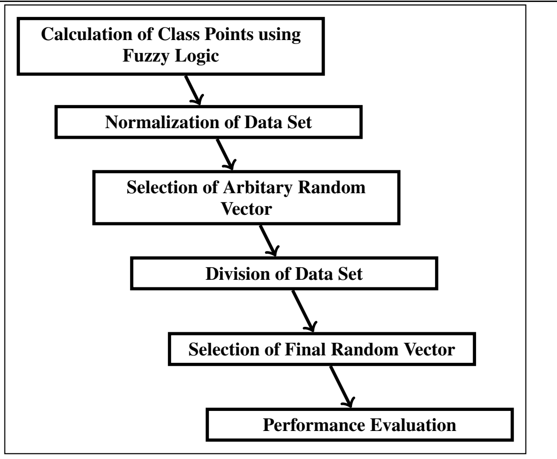
Calculation of Class Points using Fuzzy Logic
Normalization of Data Set
Selection of Arbitary Random Vector
Division of Data Set
Selection of Final Random Vector
Performance Evaluation

Figure 2: Proposed Steps Used for Effort Estimation using Random Forest Technique

Proposed Steps for Software Effort Estimation

1. **Calculation of Class Points using Fuzzy Logic**: After collecting the data from other developed projects, the CP2 value is calculated from the UML diagram. This generated CP2 value is used as an input argument. But in this study, fuzzy modeling technique has been used to calculate the weights of the various types of classes to get more accurate TUCP value. Here a *triangular membership function* is used to define their membership and compute their weights.

2. **Normalization of Dataset** : Input parameter values are normalized within the range 0 to 1. Let *S* represents the complete dataset and *s* represents a record in the *S*. Then the normalized value of *s* is obtained by using the following formula :

$$Normalized(s) = \frac{s - \min(S)}{\max(S) - \min(S)} \tag{2}$$

   where
   $\min(S)$ = min value in *S*.
   $\max(S)$ = max value in *S*.
   if $\min(S)$ is same as $\max(S)$, then Normalized(*s*) value is assigned as 0.5.

3. **Selection of Arbitrary Random Vector**: Random forest have randomness in input data and in splitting at nodes . Hence, initially an arbitrary random vector is selected to provide randomness in input data and to start the implementation process.

4. **Division of dataset**: Total no. of data are divided into two subsets i.e., training set and test set using the above arbitrary random vector.

5. **Selection of Final Random Vector**: Prediction results vary according to random vector. So an evaluation function(1- MMRE + Prediction Accuracy) is used to find a random vector. The random vector, which provides optimum value for the evaluation function is considered as final random vector.

6. **Performance Evaluation** : In this study, the Mean Magnitude of Relative Error (MMRE) and the Prediction Accuracy (PRED) are the two measures used to evaluate the performance of the model for test samples. Results obtained from proposed model are then compared with existing results to access its performance accuracy.

The above steps are followed to implement the random forest technique based effort estimation model. Finally, a comparisons of results obtained using the random forest technique based effort estimation model with the results obtained from the MLP, RBFN, SVR and SGB techniques-based models are presented to assess their performances.

5 Experimental Details

In this study, for implementing the proposed approach, dataset having forty data is being used which is also used by G. Costagliola et al. [5]. The detail description about the data set has already been provided in proposed approach section. After computing the no. of class points, the dataset is then scaled. The scaled dataset is split into two subsets i.e., *training set* and *test set*. The *training set* is used for learning reason; though the *test set* is used only for assessing the exactness of prediction of the trained model.

5.1 Calculating Class Complexity Value Using Fuzzy Logic

In the figuring of CP, Mamdani-type FIS is utilized in light of the fact that Mamdani-type FIS technique is broadly acknowledged for catching expert knowledge. It permits us to depict the expertise in more natural and human-like way. On the other hand, Mamdani-type FIS involves a significant computational load. The model has two inputs i.e. NEM and NOA; and one yield i.e. CP as demonstrated in figure 3. The primary procedures of this system incorporate four exercises: fuzzification, fuzzy rule base, fuzzy inference engine and defuzzification. All the input variables in this model are changed to the fuzzy variables focused around the fuzzification process. The complexity levels, for example, Low, Average, High, and Very High are characterized for NOA and NEM variables, for diverse number of services requested (NSR). The steps to compute the class point utilizing FIS is given below.

1. Initially develop the UML diagrams of a project and identify the number of classes in it.

2. This step deals with classification of each class into various domain such as PDT/HIT/DMT/TMT.

3. In this step, various parameters such as no. of external methods (NEM), no. of services requested (NSR) and no. of attributes (NOA) need to be extracted from UML diagram.

4. Then the complexity level such as Low, Average, High and Very High is assigned to each class focusing around their obtained NEM, NSR and NOA value.

5. During this step, the numeric value of complexity level assigned to a class is found out using fuzzy logic technique.

6. Then the Unadjusted Class Point (UACP) is calculated by multiplying the numeric value obtained in Step-5.

7. Then the TUCP is calculated by adding the UACP of all classes.

8. This step deals with calculating TCF value by using twenty four general system characteristics.

9. Finally, the adjusted CP count is calculated by multiplying TUCP with TCF.

5.1.1 Results and Discussion

A fuzzy set is defined for each linguistic value with a Triangular Membership Function (TRIMF), in figure 4. The fuzzy sets corresponding to the various associated linguistic values is defined for each type of inputs.

The proposed fuzzy guidelines hold the linguistic variables identified with the project. It is vital to note that these rules were balanced or adjusted, and in addition all pertinence level functions, as per the tests and the aspects of the project. The amount of principles those have been utilized as a part of model are 9, 16, and 9 for (0-2) NSR, (3-4) NSR and ($\geq$ 5) NSR separately for all input variables.

Fuzzy rules for (0-2)NSR:

If NEM is LOW and NOA is LOW THEN CP is LOW.
If NEM is LOW and NOA is AVERAGE THEN CP is

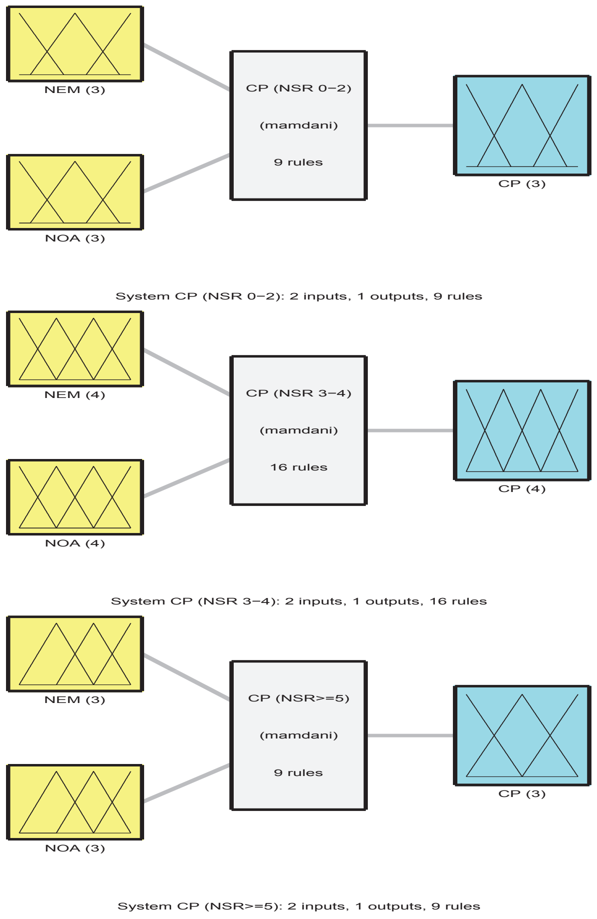
NEM (3)
NOA (3)
CP (NSR 0−2)
(mamdani)
9 rules
CP (3)
System CP (NSR 0−2): 2 inputs, 1 outputs, 9 rules
NEM (4)
NOA (4)
CP (NSR 3−4)
(mamdani)
16 rules
CP (4)
System CP (NSR 3−4): 2 inputs, 1 outputs, 16 rules
NEM (3)
NOA (3)
CP (NSR>=5)
(mamdani)
9 rules
CP (3)
System CP (NSR>=5): 2 inputs, 1 outputs, 9 rules

Figure 3: FIS for Class Point Calculation

LOW.
If NEM is LOW and NOA is HIGH THEN CP is AVERAGE.
If NEM is LOW and NOA is AVRERAGE THEN CP is AVRERAGE.
.
.
.
If NEM is HIGH and NOA is HIGH THEN CP is HIGH.

The MATLAB FIS was utilized as a part of the fuzzy computations, notwithstanding the Max-Min composition operator, the Mandani implication operator, and the Maximum operator for aggregation and after that the model is defuzzified.

Figure 5 shows the output surface of the FIS utilized for CP estimation. The fuzzy rule viewer indicated in figure 6 serves to ascertain the whole implication process from starting to end.

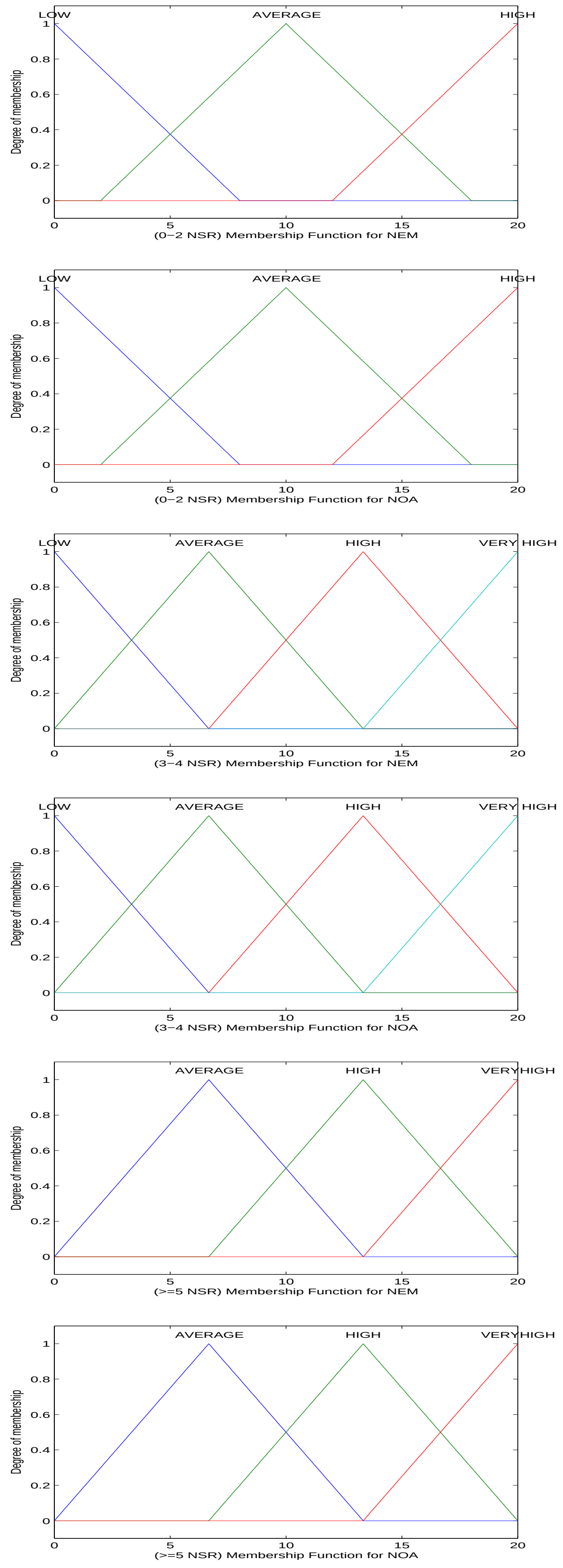

Figure 4: Triangular Graph Representation for Weight Matrix

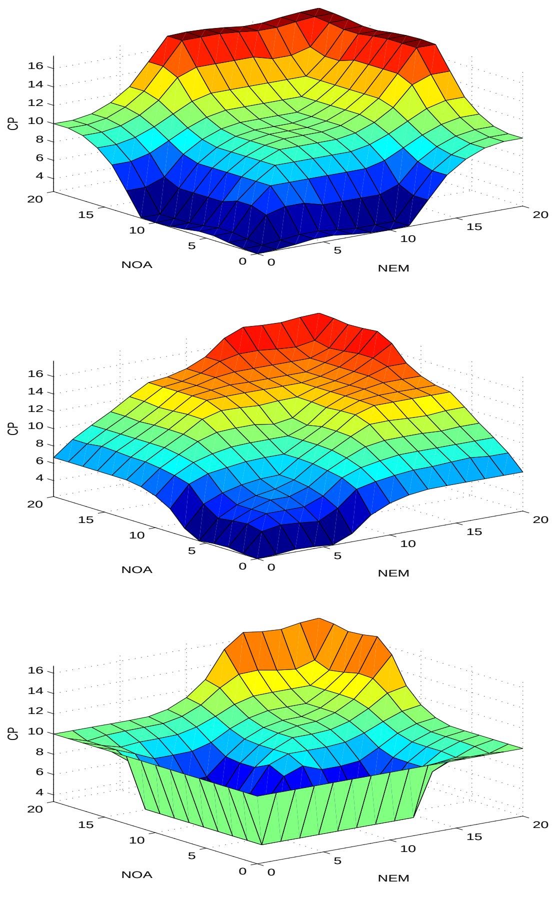

Figure 5: Surface View of Rules

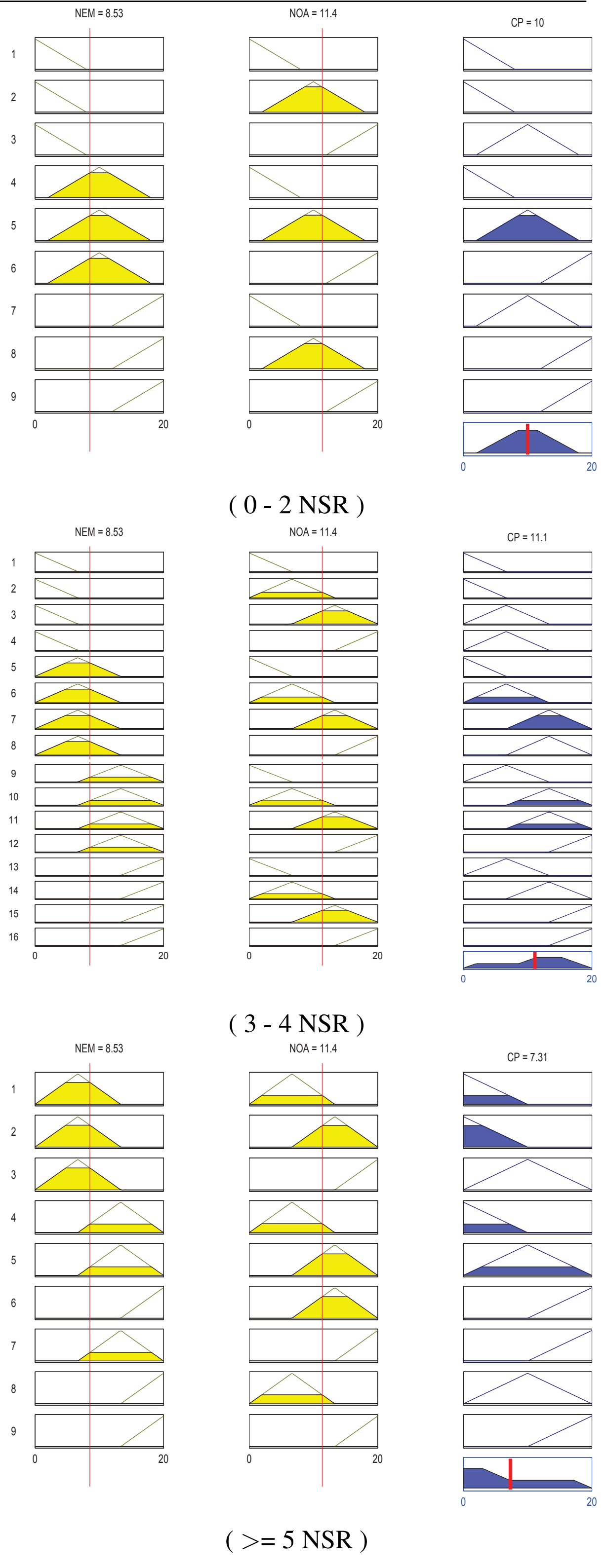

Figure 6: Fuzzy Rule View for Class Point Calculation

5.2 Model Design Using Random Forest Technique for Effort Estimation

The Brieman's algorithm is popularly used to implement the random forest technique [2]. To design an effort estimation model using the random forest technique, the following steps are used. These proposed steps help in constructing each tree, while using random forest technique.

Steps of Proposed Algorithm:

1. Let F be the number of trees in the forest. A Dataset of D points $(x_1, y_1)(x_2, y_2)....(x_D, y_D)$ is considered.
2. Each tree of the forest should be grown as follows .Steps from 3 to 9 should be repeated f times to create F number of trees.
3. Let N be the no. of training cases, and M be the no. of variables in the classifier.
4. To select training set for the tree, a random sample of n cases - yet with substitution, from the original

data of all N accessible training cases is choosen. Whatever is left of the cases are utilized to evaluate the error of the tree, by foreseeing their classes.

5. A RF tree T_f is developed to the loaded data, by repeatedly rehashing the accompanying steps for every terminal node of the tree, till the minimum node size n_{min} is arrived. Keeping in mind the end goal to make more randomness, distinctive dataset for each one trees is made.
6. The no. of input variables m is selected to ascertain the choice at a tree node. The value of m ought to be substantially short of what M.
7. For each tree node, m number of variables should be randomly chosen on which the decision at that node is based.
8. The best split focused around these m variables in the training set is calculated. The value of m ought to be held consistent throughout the development of the forest. Each tree should be fully grown and not pruned.
9. Then, the results of ensemble of trees $T_1, T_2, ..., T_f, ..., T_F$ are collected.
10. The input vector should be put down for each of the trees in the forest. In regression, it is the average of the individual tree predictions.

$$Y^F(x) = 1/F \sum_{f=1}^{F} T_f(x) \tag{3}$$

where

$Y^F(x)$ is the predicted value for the input vector x.

$T_1(x), T_2(x), ..., T_f(x)$ represents prediction value of individual trees.

There are various data objects generated by random forest technique, which needs to be considered while implementing random forest technique for software effort estimation purpose. The results obtained from these data objects need to be evaluated in order to assess the performance achieved using random forest technique.

5.2.1 The Out-Of-Bag (OOB) Error Estimate

The training set for a tree is produced by testing with substitution. During this process, something like one-third of the cases are left out of the sample. These cases are considered as out-of-bag (OOB) data. It helps in getting an impartial evaluation of the regression error value as the forest develops. OOB data also helps in getting estimation of variable importance. In RF, as the OOB is calculated internally during the run. Cross-validation of data or a different test set to obtain an impartial evaluation of the test error is not required. The computation procedure for OOB is explained below.

- During construction of each tree, an alternate bootstrap sample from the original data is used. Something like one-third of the cases from the bootstrap sample are left out and not used in the tree construction process.
- These OOB samples are put down the kth tree to obtain a regression. Using this process, a test set is acquired for each one case.
- At the end, suppose j be the predicted value that is acquired by computing the average prediction value of forest, each time case n was oob. The extent of times j is not equivalent to the actual value of n averaged over all cases is called as the *out-of-bag error estimate*.

The RF prediction accuracy can be determined from these OOB data by using the following formula.

$$OOB - MSE = \frac{1}{F} \sum_{i=1}^{F} (y_i - \bar{y}_{iOOB})^2 \tag{4}$$

where $\bar{y}_{iOOB}$ represents the average prediction value of ith observation from all trees for which this observation has been OOB. F denotes the no. of trees in the forest and y_i represents the actual value.

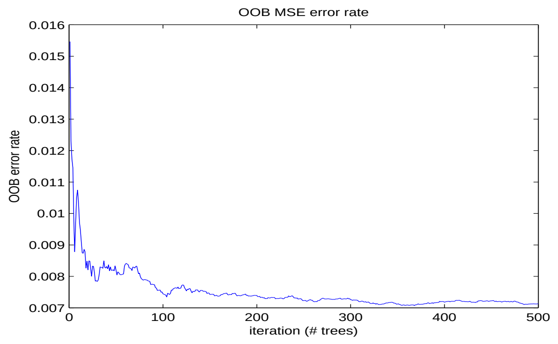

Figure 7: OOB MSE Error Rate

Figure 7 displays the OOB error rate obtained for different number of trees used in the forest. From the figure, it is quite clearly visible that during initial phase (while the number of trees used are less), the OOB error rate obtained is maximum. At the same time steadily with the increment of the amount of trees utilized within the forest, the OOB error rate converges to minimum value. After some period, OOB error rate remains constant.

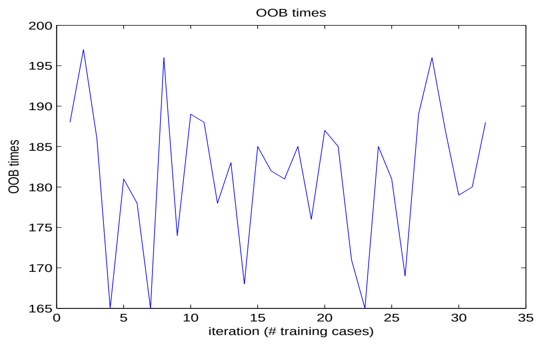

Figure 8: Number of Times Out Of Bag Occurs

Figure 8 displays the number of times, cases are out of bag for all training attributes. In this case, one hundred twenty number of training attributes are used.

5.2.2 Proximities

Proximity is one of the important data objects while calculating effort using RF technique. It measures the frequency of ending up the unique pairs of training samples in the same terminal node. It also helps in filling up the missing data in the dataset and calculating number of outliers.

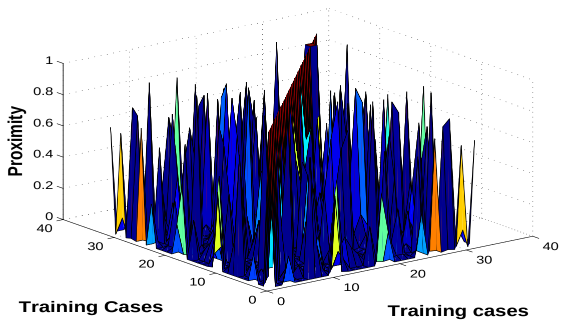

Figure 9: Proximity

Figure 9 describes the proximity value generated using random forest technique. A 120 $\times$ 120 matrix used for generating the above figure. From the figure, it is observed that, for diagonal elements, the proximity value is maximum (equals to one). But for all other elements, the proximity value is less than one. The symmetric portion adjacent to diagonal area represents other elements proximity values.

Originally, a NxN matrix is formed by the proximities. Once a tree is developed, all the data i.e., training data and out-of-bag data are put down the tree. Its proximities should be increased by one, if it is found that two cases are in the same terminal node. Finally, the normalized values of the proximities are obtained by dividing with the number of trees.

5.2.3 Complexity

In the proposed approach, 500 number of trees are taken into consideration for implementing RF technique. In the usual tree growing algorithm, all descriptors are tested for their splitting performance at each node; while Random Forest only tests m try of the descriptors. Since m try is typically very small, the search is very fast.

To get the right model complexity for optimal prediction strength, some pruning is usually done via cross validation for a single decision tree. This process can take up a significant portion of the computations. RF, on the other hand, does not perform any pruning at all. It is observed that in cases where there are an excessively large number of descriptors, RF can be trained in less time than a single decision tree. Hence, the RF algorithm can be very efficient.

5.2.4 Outlier

The cases that are expelled from the principle group of data and whose proximities to all different cases in the data mostly small are defined as *Outliers*. The concept of outliers can be revised by defining outliers relative to corresponding cases. In this way, an outlier is case whose proximities to all different cases are little. The average proximity is specified as:

$$\bar{P}(n) = \sum_{1}^{N} prox^2(n,k) \qquad (5)$$

where n and k denote a training case in the regression and N represents the total no. of training cases in the forest. The raw outlier measure for case n is specified as:

$$nsample/\bar{P}(n) \qquad (6)$$

The result of raw outlier measure inversely depends on the average proximities. The average of these raw measures and their deviations from the average are ascertained for each one cases. The final outlier measure is obtained by subtracting the average from every raw measure, and afterwards dividing it by absolute deviation.

Figure 10 describes the outlier value generated using random forest technique for 120 number of training

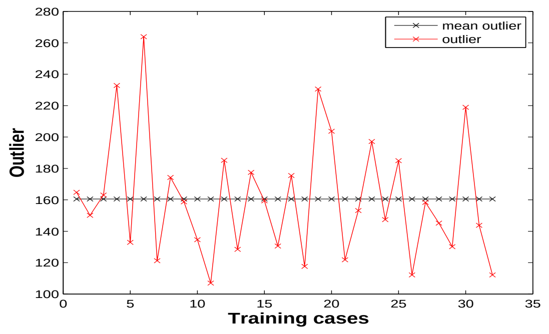

Figure 10: Outlier

cases. The outlier value is dependent on the proximity value generated using RF technique, which means that the outlier value is higher for lower proximity value and vice versa. Figure 10 displays the deviation of outlier value from the mean outlier. The training cases for which the outlier value is higher, will generate the predicted effort value deviated more from actual effort value.

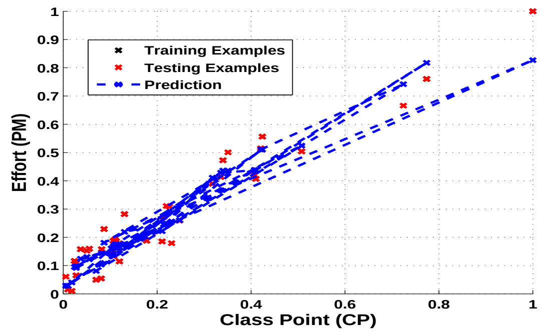

Figure 11: Random Forest Technique-based Effort Estimation Model

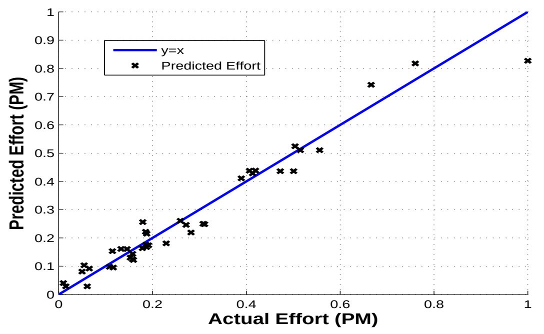

Figure 12: Variation of Predicted Effort from Actual Obtained using Random Forest Technique

This deviation is clearly visible from figure 11 and 12. Figure 11 and 12 displays the final effort estimation model obtained using RF technique. These figures show the variation of actual effort from the predicted result obtained using RF technique.

5.3 Performance Measures

The performance of the various models might be assessed by utilizing the accompanying criteria [13]:

- The **Magnitude of Relative Error (MRE)** is a very common criterion used to evaluate software cost estimation models. The MRE for each observation i can be obtained as:

$$MRE_i = \frac{|x_i - y_i|}{\bar{y}} \tag{7}$$

where
x_i = Actual Effort of i^{th} test data.
y_i = Predicted Effort of i^{th} test data.
N = Total number of data in the test set.

- The **Mean Magnitude of Relative Error (MMRE)** can be achieved through the summation of MRE over N observations

$$MMRE = \sum_{1}^{N} MRE_i \tag{8}$$

where
N = Total number of data in the test set.

- The **Prediction Accuracy (PRED)** is computed as:

$$PRED = (1 - (\frac{\sum_{i=1}^{N} |x_i - y_i|}{N})) * 100 \tag{9}$$

where
N = Total number of data in the test set.

6 Comparison

The SGB algorithm and Decision Tree Forests algorithm exhibit functional similarity, because SGB creates a tree ensemble, and also uses randomization during the creations of the trees. It creates a series of trees, and the prediction accuracy is calculated by feeding the result obtained from one tree to the next tree in the series. However, RF builds trees in parallel and also uses voting method on the prediction.

Table 1 provides a comparative study of the results obtained by some articles mentioned in the related work section. The performance of techniques used in those articles have been compared by measuring their prediction accuracy (PRED) values. Result shows that, a

Table 1: Comparison of Prediction Accuracy Values of Related Works

Sl. No.	Related Papers	Technique Used	Prediction Accuracy
1	Gennaro Costagliola et al. [5]	Regression Analysis	83%
2	Wei Zhou and Qiang Liu [25]	Regression Analysis	83%
3	S. Kanmani et al. [9]	Neural Network	87%
4	S. Kanmani et al. [10]	Fuzzy Logic	92%

maximum of 92% prediction accuracy is achieved using fuzzy logic technique. Finally, the results obtained in related work section is compared with results of proposed approaches, which is shown in table 2. The results obtained using proposed technique shows improvement in the prediction accuracy value.

Table 2: Comparison of MMRE and PRED Values between the MLP, RBFN, SVR, SGB and Random Forest Techniques

	MMRE	PRED
Multi-Layer Perceptron [20]	0.5233	94.8185%
Radial Basis Function Network [20]	0.5252	94.7539%
Support Vector Regression [22]	0.4692	95.8088%
Stochastic Gradient Boosting [19]	0.4334	95.3261%
Random Forest	0.2730	96.4250%

At the point when utilizing the MMRE, and PRED in assessment, good outcomes are entailed by lower estimations of MMRE and higher estimations of PRED. Table 2 demonstrates the comparison of MMRE and PRED values for the MLP, RBFN, SVR, SGB and RF techniques. The MLP, RBFN, SVR and SGB techniques are already been applied over class point approach with the help of same dataset as used while applying RF technique. This comparative study helps in accurately assessing the performance obtained using RF technique and proves that the results obtained using RF technique-based effort estimation model outperforms the results obtained using other existing models.

7 Conclusion

Several approaches have been considered by researchers and practitioners to calculate the effort required to develop a given software product. However, the class point model is one of the effort estimation models which is used because of its simplicity, fastness and accurateness to a certain degree. In this paper, the proposed optimized class point model has been implemented using the RF technique and generated results are compared with the results obtained from the MLP, RBFN, SVR and SGB techniques. The results demonstrate that the random forest technique provides lower estimates of MMRE and higher estimates of prediction accuracy. Consequently, it could be inferred that effort estimation utilizing the random forest technique outperforms other machine learning techniques. The computations for above procedure were implemented, and the outputs were generated using MATLAB. Extension to this procedure might be made by applying other machine learning techniques on the class point approach.

References

[1] Baskeles, B., Turhan, B., and Bener, A. Software effort estimation using machine learning methods. In *Computer and information sciences, 2007. iscis 2007. 22nd international symposium on*, pages 1–6. IEEE, 2007.

[2] Breiman, L. Random forests. *Machine learning*, 45(1):5–32, 2001.

[3] Carbone, M. and Santucci, G. Fast&&serious: a uml based metric for effort estimation. In *Proceedings of the 6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE'02)*, pages 313–322, 2002.

[4] Costagliola, G., Ferrucci, F., Tortora, G., and Vitiello, G. Towards a software size metrics for object-oriented systems. *Proc. AQUIS*, 98:121–126, 1998.

[5] Costagliola, G., Ferrucci, F., Tortora, G., and Vitiello, G. Class point: an approach for the size estimation of object-oriented systems. *Software Engineering, IEEE Transactions on*, 31(1):52–74, 2005.

[6] Elish, M. O. Improved estimation of software project effort using multiple additive regression trees. *Expert Systems with Applications*, 36(7):10774–10778, 2009.

[7] Elyassami, S. and Idri, A. Evaluating software cost estimation models using fuzzy decision trees. *Recent Advances in Knowledge Engineering and Systems Science, WSEAS Press*, pages 243–248, 2013.

[8] Huang, X., Ho, D., Ren, J., and Capretz, L. F. A neuro-fuzzy tool for software estimation. In *Software Maintenance, 2004. Proceedings. 20th IEEE International Conference on*, page 520. IEEE, 2004.

[9] Kanmani, S., Kathiravan, J., Kumar, S. S., and Shanmugam, M. Neural network based effort estimation using class points for oo systems. In *Proceedings of the International Conference on Computing: Theory and Applications*, ICCTA '07, pages 261–266, Washington, DC, USA, 2007. IEEE Computer Society.

[10] Kanmani, S., Kathiravan, J., Kumar, S. S., and Shanmugam, M. Class point based effort estimation of oo systems using fuzzy subtractive clustering and artificial neural networks. In *Proceedings of the 1st India software engineering conference*, ISEC '08, pages 141–142, New York, NY, USA, 2008. ACM.

[11] Kim, S., Lively, W., and Simmons, D. An effort estimation by uml points in early stage of software development. *Proceedings of the International Conference on Software Engineering Research and Practice*, pages 415–421, 2006.

[12] Matson, J., Barrett, B., and Mellichamp, J. Software development cost estimation using function points. *Software Engineering, IEEE Transactions on*, 20(4):275–287, 1994.

[13] Menzies, T., Chen, Z., Hihn, J., and Lum, K. Selecting best practices for effort estimation. *Software Engineering, IEEE Transactions on*, 32(11):883–895, 2006.

[14] Nassif, A. B. Enhancing use case points estimation method using soft computing techniques. *Journal of Global Research in Computer Science*, 1(4):12–21, 2010.

[15] Nassif, A. B., Capretz, L. F., and Ho, D. Estimating software effort based on use case point model using sugeno fuzzy inference system. In *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*, pages 393–398. IEEE, 2011.

[16] Nassif, A. B., Capretz, L. F., and Ho, D. Estimating software effort using an ann model based on use case points. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 42–47. IEEE, 2012.

[17] Nassif, A. B., Ho, D., and Capretz, L. F. Regression model for software effort estimation based on the use case point method. In *2011 International Conference on Computer and Software Modeling*, volume 14, pages 106–110. IACSIT Press, Singapore, 2011.

[18] Pahariya, J., Ravi, V., and Carr, M. Software cost estimation using computational intelligence techniques. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 849–854. IEEE, 2009.

[19] Satapathy, S. M., Acharya, B. P., and Rath, S. K. Class point approach for software effort estimation using stochastic gradient boosting technique. *SIGSOFT Softw. Eng. Notes*, 39(3):1–6, June 2014.

[20] Satapathy, S. M., Kumar, M., and Rath, S. K. Class point approach for software effort estimation using soft computing techniques. In *Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on*, pages 178–183. IEEE, 2013.

[21] Satapathy, S. M., Kumar, M., and Rath, S. K. Fuzzy-class point approach for software effort estimation using various adaptive regression methods. *CSI Transactions on ICT*, 1(4):367–380, 2013.

[22] Satapathy, S. M. and Rath, S. K. Class point approach for software effort estimation using various support vector regression kernel methods. In *Proceedings of the 7th India Software Engineering Conference*, ISEC '14, pages 4:1–4:10, New York, NY, USA, 2014. ACM.

[23] Sheta, A. Software effort estimation and stock market prediction using takagi-sugeno fuzzy models. In *Fuzzy Systems, 2006 IEEE International Conference on*, pages 171–178. IEEE, 2006.

[24] Sivanandam, S., Sumathi, S., Deepa, S., et al. *Introduction to fuzzy logic using MATLAB*, volume 1. Springer, 2007.

[25] Zhou, W. and Liu, Q. Extended class point approach of size estimation for oo product. In *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, volume 4, pages 117–122. IEEE, 2010.