DOI: http://dx.doi.org/10.26483/ijarcs.v8i7.4246

ISSN No. 0976-5697

Volume 8, No. 7, July – August 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

EFFORT ESTIMATION OF OBJECT ORIENTED SYSTEM USING STOCHASTIC TREE BOOSTING TECHNIQUE

Nancy Kukreja
Research Scholar
HCTM Technical Campus, Kaithal

Urvashi Garg
Assistant Professor
HCTM Technical Campus, Kaithal

Abstract: Effort Estimation is one of the necessary and daunting tasks in software engineering. Effort Estimation means to predict the effort required to develop the software project. Predicting the effort with high precision is an ultimatum that draws the concern of researchers. In a need to develop best products within proper schedule, the work of proper effort estimation is of basic necessity. No doubt, there are a lot of effort estimation techniques which are already developed like COCOMO (Cost Constructive Model) etc. but these effort estimation techniques have sustained unsuitable for estimation of object oriented software because they are used for procedural programming concept. Presently, object oriented concept is frequently used in practice and as Class is the base of object oriented design so the use of Class Point approach(CPA) to estimate the effort supports the estimator in a much better way. The performance of model obtained using CPA can be upgraded by applying Stochastic Tree Boosting (STB) technique over forty project dataset collected from different sources in order to improve its prediction accuracy.

1. INTRODUCTION

1.1 STOCHASTIC TREE BOOSTING TECHNIQUE

Stochastic Tree Boosting means to randomly select the values from dataset and then fit those values in tree in order to estimate the effort required to develop the software. Tree is a binary tree of depth 3 and each node of tree consists of values from one project of the dataset.

In this technique, firstly random number of values are selected from dataset and fitted in the first tree and values in its terminal nodes are processed. After processing the terminal nodes values in first tree again random number of values are selected and fitted in second tree and this process continues until the values are fitted in desired number of trees. After repeating this process for (number of trees) times, the values of each node are processed in order to estimate the effort of each node.

1.2 ERROR AND ACCURACY ESTIMATION METRICS

The evaluation of the values obtained using Stochastic Tree Boosting Technique is done by applying certain metrics as defined below [1-4]

The **Magnitude of Relative Error (MRE)** [1] for each observation i can be obtained as:

$$\boldsymbol{MRE_i} = \frac{|AE_i - PE_i|}{AE_i} \quad (1)$$

Where

AE_i = Original effort value collected from the dataset for the *ith* validation data.

PE_i = Output (predicted effort) obtained using the developed model for the *ith* validation data.

TP = Total no. of projects in the validation set.

The **Mean Magnitude of Relative Error (MMRE)** [1] can be obtained through the summation of Magnitude of Relative Error (MRE) over N observations:

$$\boldsymbol{MMRE} = \frac{1}{TP}\sum_{i=1}^{TP}\frac{|AE_i - PE_i|}{AE_i} \quad (2)$$

Where

AE_i = Original effort value collected from the dataset for the *ith* validation data.

PE_i = Output (predicted effort) obtained using the developed model for the *ith* validation data.

TP = Total no. of projects in the validation set.

The **Root Mean Square Error (RMSE)** [4] is calculated as the square root of mean square error (MSE). MSE is calculated by finding out the mean of the square of the difference between the actual and predicted effort value.

$$\boldsymbol{RMSE} = \sqrt{\frac{\sum_{i=1}^{TP}(AE_i - PE_i)^2}{TP}} \quad (3)$$

Where

AE_i = Original effort value collected from the dataset for the *ith* test data.

PE_i = Output (predicted effort) obtained using the developed model for the *ith* test data.

TP = Total no. of projects in the test set.

The **Prediction Accuracy (PRED (y))** [1] is PRED can be described as:

$$\boldsymbol{PRED(y)} = \left(\mathbf{1} - \left(\sum_{i=1}^{TP}\frac{|AE_i - PE_i|}{TP}\right)\right) * \mathbf{100} \quad (4)$$

Where

AE_i = Original effort value collected from the dataset for the *ith* test data.

PE_i = Output (predicted effort) obtained using the developed model for the *ith* test data.

TP = Total no. of projects in the test set.

2. LITERATURE SURVEY

The COnstructive COst Model (COCOMO) produced by Barry Boehm in 1981 [5] provides a great deal of material that explains exactly what costs the model is estimating, and why it comes up with the estimates it does.

R. T. Hughes [6] has proposed a model based on expert judgment by a group of experts to utilize their experiences for estimation of proposed software. 'Expert judgment' is defined as the consultation of one or more experts. In general, this model assumes that expert judgment is where an estimate is based on the experience of one or more people who are familiar with the development of software applications similar to that currently being sized. The Delphi technique [7] can be used to provide communication and cooperation among experts.

Function Point approach and COCOMO experience the ill effects of the impediment of the need to align the model to every individual estimation environment combined with variable precision levels even after adjustment.

Fernando Gonzalez-Ladron-de Guevara, Marta Fernandez-Diego, and Chris Lokan [8] have done a systematic mapping study over 107 number of papers that use International Software Benchmarking Standards Group (ISBSG) data to check which and to what extent variables in the ISBSG dataset have been used in software engineering to build effort estimation models.

G. Costagliola, F. Ferrucci, G. Tortora, and G. Vitiello [9] have identified two measures for calculation of final class points .i.e. Class Point 1 (CP1) and Class Point 2 (CP2) CP1 is calculated using two measures, Number of External Methods (NEM) and Number of Services Requested (NSR); whereas CP2 is calculated by utilizing an one more i.e. NOA (no of attributes) in addition to NEM and NSR. They conducted an experiment on forty project dataset and concluded that the prediction accuracy of CP1 and CP2 under the class point approach were 75% and 83% respectively.

S. Kim, W. Lively, and D. Simmons [10] have described class point in a new way to interpret system's architectural complexity. They have used various extra parameters along with NEM, NSR and NOA to compute the total number of adjusted class point value.

Shashank Mouli Satapathy, Barada Prasanna Acharya, and Santanu Kumar Rath [11] used SGB Technique for effort prediction required to develop various software projects using both the class point and the use case point approach. SGB technique considers a function iteratively in a series and combines the output of each function with a weighting coefficient in order to minimize the total error of prediction and increase the accuracy. Furthermore, he compares the models obtained using the SGB technique with the other machine learning techniques in order to highlight the performance achieved by each method.

3. PROPOSED WORK

The proposed work is based on data derived from forty student projects [9] developed using Java language. STB (Stochastic Tree Boosting) based effort estimation model which is used to estimate the effort required to develop the software has been developed using forty project dataset.

3.1 CLASS POINT APPROACH

The CPA was given by Costagliola [9] [12] in 1998. Effort Estimation process using Class point approach requires two measures CP1 (Class Point 1) and CP2 (Class Point2). CP1 is estimated using two metrics, i.e. Number of External Methods (NEM) and the Number of Services Requested (NSR); whereas CP2 is estimated utilizing a new metric NOA (No of Attributes) along with NEM and NSR. The NEM measure is given by the number of local methods which are public. NSR is basically the number of different services requested to other classes. In CP2 estimation, the Number of Attributes (NOA) [13] measure is taken into account in order to evaluate the complexity level of each class.

3.2 STEPS FOR EFFORT ESTIMATION USING STB

To calculate the effort of a given software project, the following steps are used.

1. Class Points Estimation: Class Points i.e. CP1 and CP2 are estimated using Class point approach. Estimated CP1 and CP2 values are used as an input to effort estimation.

2. Sorting of Dataset: The forty project dataset is firstly sorted in ascending order based on CP values.

3. **Setting Dataset into proper decimal**: All elements of dataset are brought into proper decimals i.e. range between 0 and 1. Let Y be the dataset and *y* be an element of the dataset. Then, the proper decimal value of y can be calculated as [4]:

$$\boldsymbol{ProperDecimal(y)} = \frac{\boldsymbol{y - min(Y)}}{\boldsymbol{max(Y) - min(Y)}} \quad (5)$$

Where min(*Y*) represents the minimum value of the dataset *Y* and max(Y) represents the maximum value of the dataset Y.

4. **Partitioning of Dataset**: The dataset is partitioned into three sets i.e. learning set, validation set and test set.

5. Performing STB Execution: The values of various parameters such as number of trees, and stochastic factor are taken and then STB steps are executed on learning set, validation set and test set.

6. Performing Validation: After completing STB execution, a five-stage validation is performed which produces five prototype models. The model that gives the minimum error (minimum RMSE and minimum MMRE) and the maximum accuracy (maximum PRED (y)) values is selected as the best model for each stage.

3.3 EXPERIMENTAL DETAILS: In the proposed research study, the dataset collected from Costagliola [9], shown in Table 1, is used. In this table, every row displays the details of one project developed in the JAVA language values of CP1, CP2 and the actual effort (denoted by EFH) expressed in terms of person-hours required to successfully complete the project.

Table 1 Forty Project Dataset [9]

S. No	EFH	CP1	CP2	NEM	NSR	NOA	S. No	EFH	CP1	CP2	NEM	NSR	NOA
1	286	103.18	110.55	142	97	170	21	366	287.97	262.74	343	264	299
2	396	278.72	242.54	409	295	292	22	947	663.6	627.6	944	421	637
3	471	473.9	446.6	821	567	929	23	485	397.1	358.6	409	269	451
4	1016	851.44	760.96	975	723	755	24	812	678.28	590.42	531	401	520
5	1261	1263.12	1242.6	997	764	1145	25	685	386.31	428.18	387	297	812
6	261	196.68	180.84	225	181	400	26	638	268.45	280.84	373	278	788
7	993	178.8	645.6	589	944	402	27	1803	2090.7	1719.25	724	1167	1633

S. No	EFH	CP1	CP2	NEM	NSR	NOA	S. No	EFH	CP1	CP2	NEM	NSR	NOA
8	552	213.3	208.56	262	167	260	28	369	114.4	104.5	192	126	177
9	998	1095	905	697	929	385	29	439	162.87	156.64	169	128	181
10	180	116.62	95.06	71	218	77	30	491	258.72	246.96	323	195	285
11	482	267.8	251.55	368	504	559	31	484	289.68	241.4	363	398	444
12	1083	687.57	766.29	789	362	682	32	481	480.25	413.1	431	362	389
13	205	59.64	64.61	79	41	98	33	861	778.75	738.7	692	653	858
14	851	697.48	620.1	542	392	508	34	417	263.72	234.08	345	245	389
15	840	864.27	743.49	701	635	770	35	268	217.36	195.36	218	187	448
16	1414	1386.32	1345.4	885	701	1087	36	470	295.26	263.07	250	512	332
17	279	132.54	74.26	97	387	65	37	436	117.48	126.38	135	121	193
18	621	550.55	481.66	382	654	293	38	428	146.97	148.35	227	147	212
19	601	539.35	474.95	387	845	484	39	436	169.74	200.1	213	183	318
20	680	489.06	438.9	347	870	304	40	356	112.53	110.67	154	83	147

After estimating the final class point values, the dataset is then sorted based on CP and then brought into proper decimal. The dataset in proper decimal format is partitioned into three sets shown in Figure 1.

3.4 PARTITIONING PROCEDURE FOR DATASET

1. Firstly, the dataset in proper decimal format is partitioned into a training set and a test set. The test set consists of every fifth tuple of dataset i.e. it consists of eight tuples and remaining thirty two tuples are present in training set. Selecting the parameters, validation and error estimation is done using training set and prediction accuracy is estimated using test set.
2. After partitioning the dataset in training set and test set, the training set is further divided into validation set and learning set. Validation set consists of every fifth tuple of training set i.e. it consists of six tuples and remaining twenty six tuples are present in learning set. Selection of parameters is done using learning set and validation and error estimation is done using validation set.

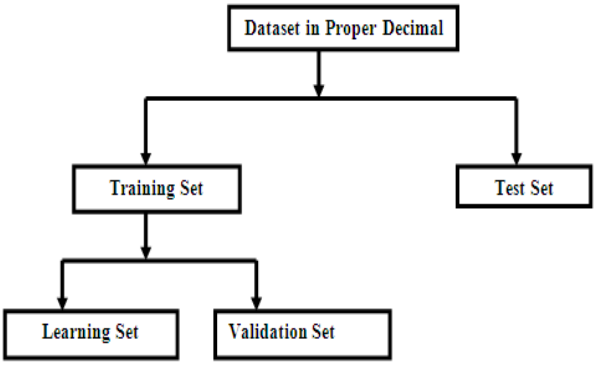


Figure 1 Divison of Dataset

3.5 STB EXECUTION

To estimate the effort required to develop the software using STB technique, the following steps are used:

1. A random percentage of rows are selected using stochastic factor. If stochastic factor is 0.5, then 50% of rows are selected.
2. The selected rows are then fitted to the first tree (T1). Each node of tree consists of its EFH value, CP value, list for containing values.
3. The mean of terminal nodes of tree is calculated.
4. The calculated mean is added to list of each terminal node of tree.
5. Steps 1-4 are again applied to feed the next tree and this process continues until 1000(number of trees) trees are fed.
6. Finally, the predicted values of each node are calculated by taking the mean of all values added to list of each node.

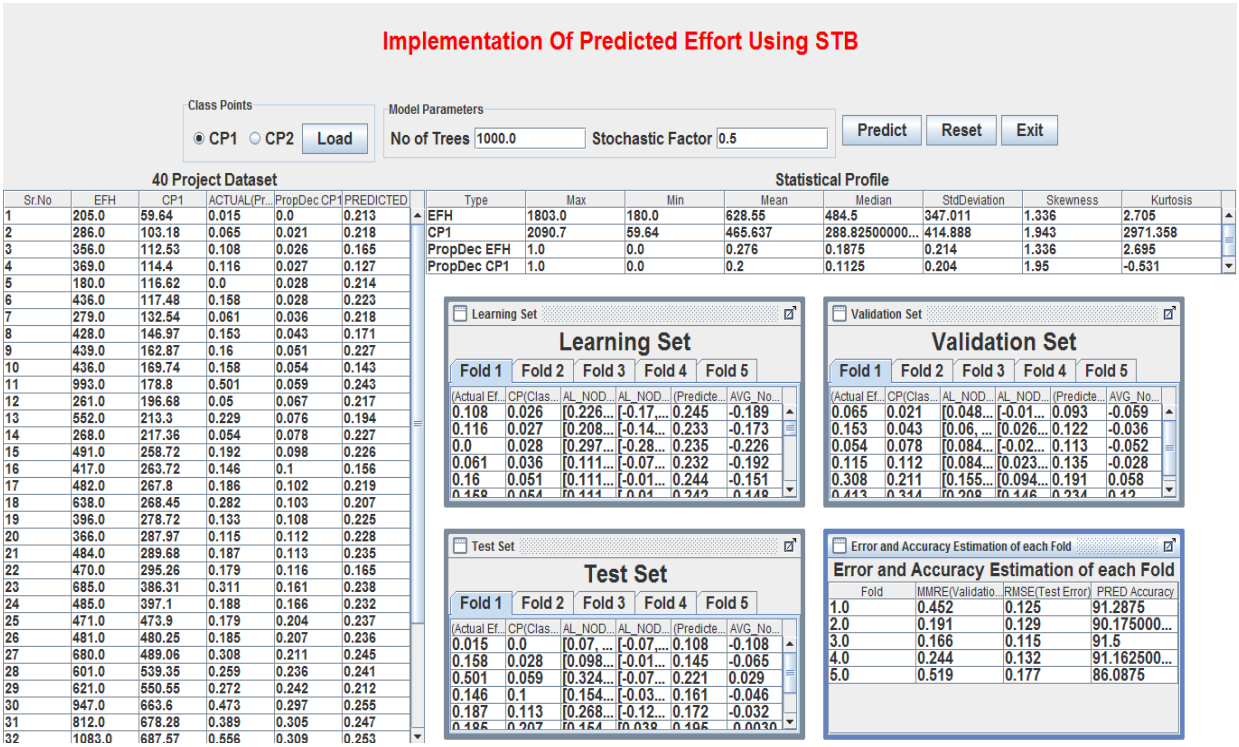
The following parameter values are chosen to predict the effort using the STB technique.
No of Trees: 1000
Stochastic Factor: 0.5

4. IMPLEMENTATION

Proper estimation of effort is very essential in order to improve reliability of software development processes. Among various estimation methods, the estimation of the effort of software is done using Class Point Approach. The parameters are optimized using the Stochastic Tree Boosting technique to achieve better accuracy. The implementation of proposed work is done using XML, Java with NetBeans IDE and MATLAB.

4.1 CALCULATING THE PREDICTED EFFORT

Figure 2 shows the Predicted Effort after completing the Effort estimation process.



Sr.No	EFH	CP1	ACTUAL(Pr...	PropDec CP1	PREDICTED
1	205.0	59.64	0.015	0.0	0.213
2	286.0	103.18	0.065	0.021	0.218
3	356.0	112.53	0.108	0.026	0.165
4	369.0	114.4	0.116	0.027	0.127
5	180.0	116.62	0.0	0.028	0.214
6	436.0	117.48	0.158	0.028	0.223
7	279.0	132.54	0.061	0.036	0.218
8	428.0	146.97	0.153	0.043	0.171
9	439.0	162.87	0.16	0.051	0.227
10	436.0	169.74	0.158	0.054	0.143
11	993.0	178.8	0.501	0.059	0.243
12	261.0	196.68	0.05	0.067	0.217
13	552.0	213.3	0.229	0.076	0.194
14	268.0	217.36	0.054	0.078	0.227
15	491.0	258.72	0.192	0.098	0.226
16	417.0	263.72	0.146	0.1	0.156
17	482.0	267.8	0.186	0.102	0.219
18	638.0	268.45	0.282	0.103	0.207
19	396.0	278.72	0.133	0.108	0.225
20	366.0	287.97	0.115	0.112	0.228
21	484.0	289.68	0.187	0.113	0.235
22	470.0	295.26	0.179	0.116	0.165
23	685.0	386.31	0.311	0.161	0.238
24	485.0	397.1	0.188	0.166	0.232
25	471.0	473.9	0.179	0.204	0.237
26	481.0	480.25	0.185	0.207	0.236
27	680.0	489.06	0.308	0.211	0.245
28	601.0	539.35	0.259	0.236	0.241
29	621.0	550.55	0.272	0.242	0.212
30	947.0	663.6	0.473	0.297	0.255
31	812.0	678.28	0.389	0.305	0.247
32	1083.0	687.57	0.556	0.309	0.253

Type	Max	Min	Mean	Median	StdDeviation	Skewness	Kurtosis
EFH	1803.0	180.0	628.55	484.5	347.011	1.336	2.705
CP1	2090.7	59.64	465.637	288.82500000...	414.888	1.943	2971.358
PropDec EFH	1.0	0.0	0.276	0.1875	0.214	1.336	2.695
PropDec CP1	1.0	0.0	0.2	0.1125	0.204	1.95	-0.531

Figure 2 Calculating the Predicted Effort in Predicted Effort Frame based on CP1

4.2 CALCULATED RESULTS OF EFFORT ESTIMATION

Table 2 Error and Accuracy Estimation for Each Fold obtained using STB based on CP1

Fold	MMRE(Validation Error)	RMSE(Test Error)	Prediction Accuracy (%)
1	0.452	0.125	91.2875
2	0.191	0.129	90.175000
3	0.166	0.115	91.5
4	0.244	0.132	91.162500
5	0.519	0.177	86.0875

Table 2 shows that 3rd Fold is the best fold as it provides min MMRE, min RMSE and max Prediction accuracy.

Table 3 Final Estimated/Predicted Effort using STB based on CP1

S. No	EFH	CP1	Proper Decimal EFH (Actual Effort)	Proper Decimal CP1	Predicted Effort	S. No	EFH	CP1	Proper Decimal EFH (Actual Effort)	Proper Decimal CP1	Predicted Effort
1.	205	59.64	0.015	0	0.213	21.	484	289.68	0.187	0.113	0.235
2.	286	103.18	0.065	0.021	0.218	22.	470	295.26	0.179	0.116	0.165
3.	356	112.53	0.108	0.026	0.165	23.	685	386.31	0.311	0.161	0.238
4.	369	114.4	0.116	0.027	0.127	24.	485	397.1	0.188	0.166	0.232
5.	180	116.62	0	0.028	0.214	25.	471	473.9	0.179	0.204	0.237
6.	436	117.48	0.158	0.028	0.223	26.	481	480.25	0.185	0.207	0.236
7.	279	132.54	0.061	0.036	0.218	27.	680	489.06	0.308	0.211	0.245
8.	428	146.97	0.153	0.043	0.171	28.	601	539.35	0.259	0.236	0.241
9.	439	162.87	0.16	0.051	0.227	29.	621	550.55	0.272	0.242	0.212
10.	436	169.74	0.158	0.054	0.143	30.	947	663.6	0.473	0.297	0.255
11.	993	178.8	0.501	0.059	0.243	31.	812	678.28	0.389	0.305	0.247
12.	261	196.68	0.05	0.067	0.217	32.	1083	687.57	0.556	0.309	0.253
13.	552	213.3	0.229	0.076	0.194	33.	851	697.48	0.413	0.314	0.278

S. No	EFH	CP1	Proper Decimal EFH (Actual Effort)	Proper Decimal CP1	Predicted Effort	S. No	EFH	CP1	Proper Decimal EFH (Actual Effort)	Proper Decimal CP1	Predicted Effort
14.	268	217.36	0.054	0.078	0.227	34.	861	778.75	0.42	0.354	0.25
15.	491	258.72	0.192	0.098	0.226	35.	1016	851.44	0.515	0.39	0.29
16.	417	263.72	0.146	0.1	0.156	36.	840	864.27	0.407	0.396	0.258
17.	482	267.8	0.186	0.102	0.219	37.	998	1095	0.504	0.51	0.273
18.	638	268.45	0.282	0.103	0.207	38.	1261	1263.1	0.666	0.593	0.397
19.	396	278.72	0.133	0.108	0.225	39.	1414	1386.3	0.76	0.653	0.284
20.	366	287.97	0.115	0.112	0.228	40.	1803	2090.7	1	1	0.316

Table 3 shows the Final Predicted Effort of forty project dataset.

4.3 SOFTWARE SIZE VS. EFFORT GRAPH BASED ON CP1

Figure 3 depicts the relationship between Software Size (Class Points) and Actual Effort.

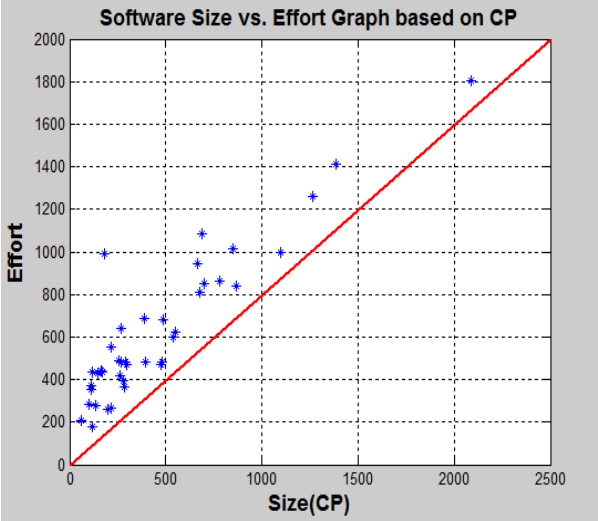


Figure 3 Software Size vs. Effort Graph based on CP1

4.4 SOFTWARE SIZE VS. EFFORT GRAPH (PROPER DECIMAL) BASED ON CP1

Figure 4 depicts the relationship between Software Size(Proper Decimal) and Effort(Proper Decimal) based on CP1. Proper Decimal means that values lies between 0 and 1

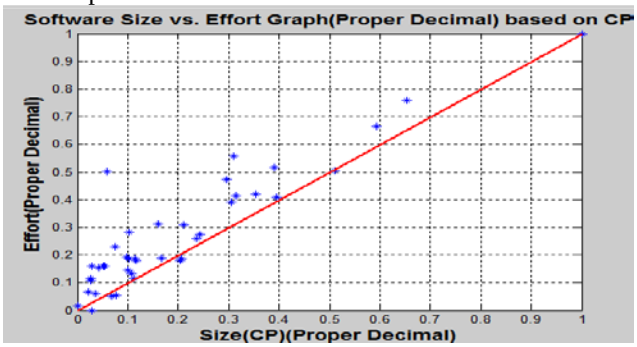


Figure 4 Software Size vs. Effort Graph (Proper Decimal) based on CP1 `

4.5 ACTUAL EFFORT VS. PREDICTED EFFORT GRAPH BASED ON CP1

Figure 5 depicts the relationship between Actual Effort and Predicted Effort. It shows that Predicted Effort and Actual Effort are somewhat close to each other i.e. there is little difference between the values of Actual effort and predicted effort.

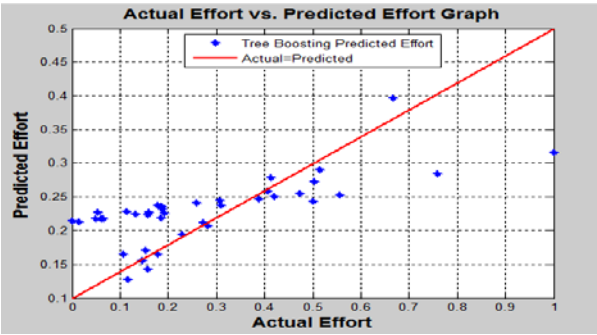


Figure 5 Actual Effort vs. Predicted Effort Graph based on CP1

5. CONCLUSIONS

The research work proposed in this research paper is beneficial for software developers, system analysts, and product experts. Class Point Approach (CPA) is used for object-oriented software and I have extended this approach by employing Stochastic Tree Boosting Technique to provide more precise estimation result. I have observed different results (error and prediction accuracy values) obtained using STB and comparisons are made using graphs. The results show that the STB-based effort estimation model possesses lower MMRE, NRMSE and higher prediction accuracy. So, we can conclude that effort estimation using the STB-based model provides results with better precision.

6. FUTURE SCOPE OF WORK

This research work can be further extended by applying some other machine learning techniques for the software development effort estimation purpose. There are various machine learning methods such as Decision Tree Forest, Random Forest and Support Vector Regression etc. which

can be implemented and compare their results with the results of the STB technique to measure their precision.

7. REFERENCES

[1] Tim Menzies, Zhihao Chen, Jairus Hihn, and Karen Lum, "Selecting best practices for effort estimation," Software Engineering, IEEE Transactions on, 32(11):883-895, 2006.

[2] Ali Bou Nassif, Danny Ho and Luiz Fernando Capretz, "Towards an early software estimation using log-linear regression and a multilayer perceptron model," Journal of Systems and Software, 86(1):144-160, 2013.

[3] Mohammad Azzeh, Ali Bou Nassif, and Leandro L Minku, "An empirical evaluation of ensemble adjustment methods for analogy-based effort estimation," Journal of Systems and Software, 103:36-52, 2015.

[4] Shashank Mouli Satapathy, Mukesh Kumar, and Santanu Kumar Rath, "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.

[5] Barry W Boehm, "Software engineering economics, volume 197," Prentice-hall Englewood Cliffs (NJ), 1981.

[6] Robert T Hughes, "Expert judgment as an estimating method," Information and Software Technology, 38(2):67-75, 1996.

[7] Norman Dalkey and Olaf Helmer, "An experimental application of the delphi method to the use of experts," Management science, 9(3):458-467, 1963.

[8] Fernando Gonzalez-Ladron-de Guevara, Marta Fernandez-Diego, and Chris Lokan, "The usage of isbsg data fields in software effort estimation: A systematic mapping study," Journal of Systems and Software, 113:188-215, 2016.

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

[10] S. Kim, W. Lively and D. Simmons, "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.

[11] Shashank Mouli Satapathy, Barada Prasanna Acharya, and Santanu Kumar Rath, "Class point approach for software effort estimation using stochastic gradient boosting technique," ACM SIGSOFT Software Engineering Notes, 39(3):1-6, 2014.

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

[13] B. Henderson-Sellers, "Object-Oriented Metrics: Measures of Complexity," Prentice-Hall, 1996.