

Effort Estimation For Object-oriented System
Using
Stochastic Gradient Boosting Technique

Barada prasanna Acharya



Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India
May 2014

Effort Estimation For Object-oriented System Using Stochastic Gradient Boosting Technique

Thesis submitted in partial fulfilment of the requirements for the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Barada prasanna Acharya

(Roll- 110CS0111)

Under the supervision of

Prof. S. K. Rath



Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, 769 008, India

May 2014



Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India.

Certificate

This is to certify that the work in the thesis entitled *Effort Estimation For Object-oriented System Using Stochastic Gradient Boosting Technique* by *Barada prasanna Acharya* is a record of an original research work carried out by him under my supervision and guidance in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in the department of Computer Science and Engineering, National Institute of Technology Rourkela. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Place: NIT Rourkela
Date: May 12, 2014

(Prof. Santanu Ku. Rath)
Professor, CSE Department
NIT Rourkela, Odisha



Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India.

Authors Declaration

I hereby declare that all the work contained in this report is my own work unless otherwise acknowledged. Also, all of my work has not been previously submitted for any academic degree. All sources of quoted information have been acknowledged by means of appropriate references.

Place: NIT Rourkela
Date: May 12, 2014

(Barada prasanna Achary)
CSE Department
NIT Rourkela, Odisha

Acknowledgment

I am grateful to numerous local and global peers who have contributed towards shaping this thesis. At the outset, I would like to express my sincere thanks to Prof. Santanu Ku. Rath for his advice during my thesis work. As my supervisor, he has constantly encouraged me to remain focused on achieving my goal. His observations and comments helped me to establish the overall direction to the research and to move forward with investigation in depth. He has helped me greatly and been a source of knowledge.

I would like to thank Mr. Shashank Mouli Satapathy for his encouragement and support, without which this project could not have seen the light of the day.

I am obliged to the faculty members of the Department of Computer Science and Engineering at NIT Rourkela for the valuable information provided by them in their respective fields. I must acknowledge the academic resources that I have got from NIT Rourkela.

I would like to thank all my friends and lab-mates for their encouragement and understanding. Their help can never be penned with words.

Last, but not the least, I would like to dedicate this thesis to my family, for their love, patience and understanding.

Barada prasanna Acharya

Roll-110CS0111

Abstract

The success of software development depends on the proper prediction of the effort required to develop the software. Project managers oblige a solid methodology for software effort prediction. It is particularly paramount throughout the early stages of the software development life cycle. Faultless software effort estimation is a major concern in software commercial enterprises. Stochastic Gradient Boosting (SGB) is a machine learning techniques that helps in getting improved estimated values. SGB is used for improving the accuracy of estimation models using decision trees. In this paper, the basic aim is the effort prediction required to develop various software projects using both the class point and the use case point approach. Then, optimization of the effort parameters is achieved using the SGB technique to obtain better prediction accuracy. Furthermore, performance comparisons of the models obtained using the SGB technique with the other machine learning techniques are presented in order to highlight the performance achieved by each method.

Keywords: Class Point Approach, Object-oriented Analysis and Design, Software Effort Estimation, Stochastic Gradient Boosting, Use case point Approach.

Contents

Certificate	ii
Authors Declaration	iii
Acknowledgement	iv
Abstract	v
List of Figures	viii
List of Tables	ix
1 Introduction	1
1.1 Basic Concepts	2
1.1.1 Top-down Approach	2
1.1.2 Bottom-up Approach	2
1.2 Class Point Analysis	4
1.3 Use Case Point Analysis	7
1.4 Various Performance Measures	10
1.4.1 Mean Square Error (MSE)	11
1.4.2 Magnitude of Relative Error (MRE)	11
1.4.3 Mean Magnitude of Relative Error (MMRE)	11
1.4.4 Root Mean Square Error (RMSE)	11
1.4.5 Normalized Root Mean Square(NRMS)	11
1.4.6 Prediction Accuracy (PRED)	11
1.5 Dataset used for Effort Calculation	12
1.6 Thesis Organization	12
2 Literature Survey	13

3	Stochastic Gradient Boosting Technique	16
3.1	Introduction	16
3.2	Stochastic Gradient Boosting Technique	16
3.3	Proposed Approach	17
3.4	Experimental Details	19
3.5	Model Design Using Stochastic Gradient Boosting Technique	20
3.6	Summary	21
4	Results and Comparison	22
4.1	Class Point Approach Result	22
4.1.1	Comparison	23
4.2	Use Case Point Approach Result	25
4.2.1	comparison	26
4.3	Summary	28
5	Conclusion and Future Work	29
	Bibliography	30
	Dissemination of Work	34

List of Figures

1.1	Phases for Class Point Calculation	5
1.2	Phases for Calculation of Use Case Point	7
3.1	Proposed Steps Used for Effort Estimation using SGB	19
4.1	The SGB-based Effort Estimation Model for class point approach	23
4.2	Actual vs. Predicted Effort using SGB for class point approach	23
4.3	Comparison of MMRE and PE Values	24
4.4	Actual vs. Predicted Effort using SGB for use case point	26
4.5	The SGB-based Effort Estimation Model for use case point	26

List of Tables

1.1	Evaluation of Complexity Level for CP1	5
1.2	Evaluation of Complexity Level for CP2	6
1.3	TUCP evaluation for Each Class Type	6
1.4	DI of Twenty Four General System Characteristics	7
1.5	Weighting Factors of Actor	8
1.6	Weighting Factors Use Case	8
1.7	Technical Factor values	9
1.8	Environment Factors	10
4.1	MMRE and NRMSE Values Obtained using SGB for Each Fold	22
4.2	Comparison of Efforts obtained using MLP, RBFN and SGB	24
4.3	Comparison of Prediction Accuracy Values of Related Works	25
4.4	Comparison of Efforts obtained using MLP, RBFN and SGB	25
4.5	Comparison of Prediction Accuracy Values of Related Works	27
4.6	Comparison of MMRE, NRMSE and PRED Values	27

Chapter 1

Introduction

According to survey, more than one-third projects outmatch their budget and suffer late delivery and about two-thirds of all projects run over their initial estimates. The basic reasons for above problems are the inability of a project manager or system analyst to predict the cost and effort required to develop a software accurately. Without accurate effort estimation capability, project managers face difficulty in determining how much time and manpower the project should take to be finished.

Software effort estimation is always a difficult task to understand and estimate in initial phases of software development life cycle, as a software product cant be seen and touched. Software grow and change when it is written. In project management, the most challenging task is effort estimation. Need for precise estimation require assets and timetables for programming improvement ventures. By and large Software cost estimation procedure incorporates the accompanying :

- Estimation of the size of the software product is required
- Estimation of the effort prediction
- Development of project schedules
- Estimation of cost of the project

First step towards estimation involves understanding and defining the system to be estimated. A software is intractable and invisible during its early stages of production. Understanding and estimating a product or process which cannot be seen and touched is often more difficult and challenging. Often hardware

design failure leads towards change in the software. Late changes during the development process results in unanticipated software growth. After several years research, there exist many software cost estimation methods like, estimation by analogy, algorithmic methods, expert judgement method, bottom-up method and top-down method. As weaknesses and strengths of these methods are often complimentary, we can not say a method is better or worse than the other. Before estimation of projects, understanding the strengths and weaknesses of every method is important.

1.1 Basic Concepts

1.1.1 Top-down Approach

In top-down approach, generally procedure-oriented software designing approach is used. In this type of approach, sub-systems are occurred by breaking down the system into its compositional and an overview of the system is generated, which specifies but does not give detail about any first-level subsystems. Each subsystem is again elaborated in greater detail. Macro Model is also a name of top-down estimating method. Using this method, using the globally defined properties of the software project, an overall project cost estimation is derived, and then the project is partitioned into many low-level components. In early phase of software development life cycle, this methodology is extremely helpful, in light of the fact that at first there are no point by point data accessible.

1.1.2 Bottom-up Approach

Bottom-up Approach is popularly known as object-oriented programming concept. Several features such as inheritance, encapsulation, abstraction and polymorphism, offered by object-oriented programming play important roles in the development of software products. Size estimation of a software product is one of the most important measures. Conventional software estimation techniques like Function Point Analysis (FPA) and Constructive Cost Estimation Model (CO-COMO) are not suitable for precise measurement of the effort and cost of all types

of software. It is also observed that the line of code (LOC) and function point (FP) methods are both adopted from procedural programming [1]. Procedure-oriented design divides the data and procedure; in case of object-oriented concepts merges them. The extent that effort estimation is concerned, various unsolved issues regardless blunders exist. Correct estimation of a software project is constantly critical for deciding the achievability of the undertaking regarding examination of expense-profit [2].

In the present scenario, most software project planning depends upon accurate effort estimation. Since the fundamental intelligent unit of an object-oriented system is class, more accurate result can be generated by using the class point approach to predict the effort required for a project. For the effort estimation process using class point, two measures, CP1 and CP2, are used. CP1 is computed using two metrics, the former one is Number of External Methods (NEM) and the later one is Number of Services Requested (NSR); whereas CP2 is computed by using a new metric along with NEM and NSR, called as Number of Attributes (NOA). The size of the interface of a class, which is decided by the no. of local public methods is measured by NEM; whereas NSR offers a measurement for the interconnection between the system components. On the other hand, NOA helps in finding out the number of attributes used in a class. For both the function point analysis (FPA) as well as the class point approach (CPA), the computation of the Technical Complexity Factor (TCF) is dependant upon the influence of general system characteristics. However, in the class point approach, non-technical factors such as Developer's professional competence, Management efficiency, Reliability, Security, Portability and Maintainability are not taken into account [3].

The Use Case Point (UCP) model estimates the effort of a software product by using use case diagrams. UCP helps in providing more accurate effort estimation from design phase of software development life cycle. The no. of use cases and the no. of actors are counted for the measurement of UCP, which are multiplied by complexity factors assigned with them. Use cases and actors are consists of three categories, such as average, simple and complex. The complexity value

(complex, average or simple) determination of use cases are calculated by the no. of transactions used per use case.

Hence in this paper, both the class point approach and the use case point method are utilised for the calculation the effort needed to create a software product item and and their performances are analysed. In order to get better prediction accuracy, a Stochastic Gradient Boosting based effort estimation model is used. The results obtained from the SGB-based estimation model are then compared with the results obtained from a Multi Layer Perception (MLP) and a Radial Basis Function Network (RBFN) based models.

1.2 Class Point Analysis

Gennaro Costagliola et al. has introduced the class point approach in 1998 [4]. Based on the function point analysis approach, The idea using the *Class Point Approach* is to quantify the classes. Functions or procedures are the basic programming units in procedure-oriented model; whereas, in case of an object-oriented model, the fundamental building blocks are classes. The Class Point size estimation process consists of three basic phases, corresponding to similar phases in the function point approach [5] :

- Information processing size prediction:
 - Classes are identified and classified
 - Complexity level of each class are assigned
 - Total Unadjusted Class Point is estimated
- Estimation Technical complexity factor
- Evaluation of Final Class Points

In the beginning step, the requirement specifications of the software's design are analysed for identification and classification of the classes into four different types of system components. They are Human Interaction Type (HIT), Problem

Domain Type (PDT), Task Management Type (TMT) and Data-Management Type (DMT) [5].

In the second step of first phase, each above identified class is given a complexity level which is based on the local declared methods of the class and of the association of the comparing class with whatever remains of the system. For CP1, the complexity level for individual class is evaluated based on the No. of External Methods (NEM), and the No. of Services Requested (NSR). Similarly for CP2, to evaluate the complexity level of each class, the No. of Attributes (NOA) measure is considered with the above measures. The following block diagram, which is shown in Figure-1.1 describes the steps to compute the class point.

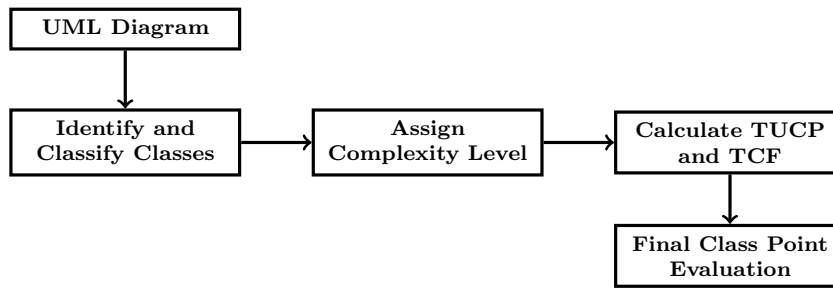


Figure 1.1: Phases for Class Point Calculation

In case of CP1 calculation, the complexity level for the class is evaluated according to Table-1.1 depending on the value of NSR and NEM [5]. For example, a class having NSR value 3 and NEM value 7, average complexity level will be assigned to the corresponding class.

Table 1.1: Evaluation of Complexity Level for CP1

	0 - 4 NEM	5 - 8 NEM	9 - 12 NEM	≥ 13 NEM
0 - 1 NSR	Low	Low	Average	High
2 - 3 NSR	Low	Average	High	High
4 - 5 NSR	Average	High	High	Very High
> 5 NSR	High	High	Very High	Very High

In case of CP2 calculation, the complexity level for a class is evaluated using NEM, NOA and NSR values corresponding to Table- 1.2a, 1.2b and 1.2c [5]. In these tables, NEM and NOA range vary according to the fixed NSR range.

After assignment of complexity level for every class, weight to the class is assigned based on information and its type which are given in Table- 1.3 [5]. After

Table 1.2: Evaluation of Complexity Level for CP2

<i>0 - 2 NSR</i>	<i>0 - 5 NOA</i>	<i>6 - 9 NOA</i>	<i>10 - 14 NOA</i>	≥ 15 NOA
<i>0 - 4 NEM</i>	Low	Low	Average	High
<i>5 - 8 NEM</i>	Low	Average	High	High
<i>9 - 12 NEM</i>	Average	High	High	Very High
≥ 13 NEM	High	High	Very High	Very High

(a)

<i>3 - 4 NSR</i>	<i>0 - 4 NOA</i>	<i>5 - 8 NOA</i>	<i>9 - 13 NOA</i>	≥ 14 NOA
<i>0 - 3 NEM</i>	Low	Low	Average	High
<i>4 - 7 NEM</i>	Low	Average	High	High
<i>8 - 11 NEM</i>	Average	High	High	Very High
≥ 12 NEM	High	High	Very High	Very High

(b)

≥ 5 NSR	<i>0 - 3 NOA</i>	<i>4 - 7 NOA</i>	<i>8 - 12 NOA</i>	≥ 13 NOA
<i>0 - 2 NEM</i>	Low	Low	Average	High
<i>3 - 6 NEM</i>	Low	Average	High	High
<i>7 - 10 NEM</i>	Average	High	High	Very High
≥ 11 NEM	High	High	Very High	Very High

(c)

that, the Total Unadjusted Class Point value (TUCP) is calculated as a weighted sum of the no. of classes of different component types.

$$TUCP = \sum_{i=1}^4 \sum_{j=1}^3 w_{ij} \times x_{ij} \quad (1.1)$$

where x_{ij} is the no. of classes of component type i (human interaction, task management, problem domain etc.) with the complexity level j (low, average or high), For i type and j complexity level, w_{ij} is the weighting value.

Table 1.3: TUCP evaluation for Each Class Type

System Component Type	Description	Complexity			
		Low	Average	High	Very High
PDT	Problem Domain Type	3	6	10	15
HIT	Human Interaction Type	4	7	12	19
DMT	Data Management Type	5	8	13	20
TMT	Task Management Type	4	6	9	13

The Total Degree of Influence (TDI) is calculated by the aggregate of the influence degrees of all the general system characteristics [5], which is shown in Table-1.4. This is used in determining the TCF according to the below formula:

$$TCF = 0.55 + (0.01 * TDI) \quad (1.2)$$

Finally, the Class Point (CP) value is determined by multiplying the Total Unadjusted Class Point (TUCP) value by TCF.

$$CP = TUCP * TCF \quad (1.3)$$

Table 1.4: DI of Twenty Four General System Characteristics

ID	System Characteristics	DI	ID	System Characteristics	DI	
C1	Data Communication	C13	Multiple sites	
C2	Distributed Functions	C14	Facilitation of change	
C3	Performance	C15	User Adaptivity	
C4	Heavily used configuration	C16	Rapid Prototyping	
C5	Transaction rate	C17	Multi-user Interactivity	
C6	On-line data entry	C18	Multiple Interfaces	
C7	End-user efficiency	C19	Management Efficiency	
C8	On-line update	C20	Developers' Professional Competence	
C9	Complex processing	C21	Security	
C10	Re-usability	C22	Reliability	
C11	Installation ease	C23	Maintainability	
C12	Operational ease	C24	Portability	
TDI	Total Degree of Influence (TDI)				

The final class point of various projects is used for calculation of the required effort to develop the project in a very scheduled time.

1.3 Use Case Point Analysis

The Use Case Point (UCP) model was proposed by Gustav Karner in 1993 [6]. This system is accomplished by developing Function Point Analysis and Mk II Function Point Analysis, based on the rationality of above strategies. An early use case based software effort estimation might be made when there exist some information of the system size, architectural planning and issue area at the phase of the estimation. The block diagram, which is shown in Figure 1.2 describes the steps to calculate the class point.

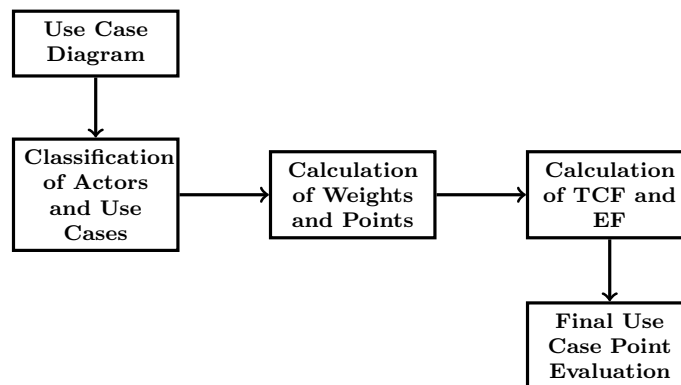


Figure 1.2: Phases for Calculation of Use Case Point

The use case point approach can be implemented using the following steps :

Classification of Actors and Use Cases

The first step is about the classification of the actors into classes like complex, average or simple. A simple actor is represented by a system which has a definite Application Programming Interface, (API). For an average actor, it is an system, which is interacted through a protocol and in case of a complex actor, it may be a person who interacts through a Web page or a GUI. Each actor type is assigned a weighting factor, in the following manner :

Table 1.5: Weighting Factors of Actor

Actor Type	Weighting Factor
Simple	1
Average	2
Complex	3

Similarly every use case is classified as average, simple or complex which depends on no. of transactions between cases in the use case description diagram and include secondary scenarios. Set of activities is called as a transaction which is either executed completely, or not at all. No. of transactions is counted by counting the number of use case steps. The use case is considered as *Simple* type, when a simple user interface is used by it and interacts with only a single database entity. The use case is considered as *Average* type, when it uses more than one interface design and interacts with 2 or more database entities. Similarly the use case is *Complex* type, when it uses a complex user interface or processing and interacts with 3 or more database entities. Then use case complexity is defined and weighted in the below described manner:

Table 1.6: Weighting Factors Use Case

Use Case Type	No. of Transactions Type	Weighting Factor
Simple	≤ 3	5
Average	4 to 7	10
Complex	≥ 7	15

Calculation of Weights and Points

The total Unadjusted Actor Weights (UAW) is measured by counting number of actors of each kind (by degree of complexity), then multiplied each total with weighting factor, and finally the products are added up. Each type of use case is then multiplied with weighting factor and the generated products are added up to find the Unadjusted Use Case Weights (UUCW). Then the UAW is summed up to the UUCW to get the Unadjusted Use Case Points (UUCP).

$$UUCP = UAW + UUCW \quad (1.4)$$

Calculation of TCF and EF

The UUCP are calculated based on the values of a number of technical and environmental factors which are shown in Tables 1.7 and 1.8. Each factor is scaled a value between 0 and 5, depends on its influence on the project. Rating of 0 signifies the factor has no impact upon this project and 5 signifies it is very essential for this project.

Table 1.7: Technical Factor values

Factor	Description	Weight
T1	Distributed System	2
T2	Response Adjectives	2
T3	End-user Efficiency	1
T4	Complex Processing	1
T5	Reusable Code	1
T6	Easy to Install	0.5
T7	Easy to Use	0.5
T8	Portable	2
T9	Easy to Change	1
T10	Concurrent	1
T11	Security Features	1
T12	Access for Third Parties	1
T13	Special Training Required	1

The adjusted use case points are produced by the multiplication of adjustment factors by the unadjusted use case points, which yield an estimation of the size of the software. The Technical Complexity Factor (TCF) is computed by multiplying each factor value (T1- T13) with its weight and then summing all these numbers

Table 1.8: Environment Factors

Factor	Description	Weight
F1	Familiar with RUP	1.5
F2	Application Experience	0.5
F3	Object-oriented Experience	1
F4	Lead Analyst Capability	0.5
F5	Motivation	1
F6	Stable Requirements	2
F7	Part-time Workers	-1
F8	Difficult Programming Language	2

to find the summation called the *TFactor*. The below formula is applied to find TCF:

$$TCF = 0.6 + (0.01 * TFactor) \quad (1.5)$$

The Environmental Factor (EF) is computed by the multiplication of each factor (F1-F8) with its weight and summing the products to get the summation called the *EFactor*. The below equation gives EF value:

$$EF = 1.4 + (-0.03 * EFactor) \quad (1.6)$$

Final Use Case Point Calculation

The final adjusted use case points (UCP) are calculated as follows:

$$UCP = UUCP * TCF * EF \quad (1.7)$$

The final use case point value is then considered as input argument to SGB model to calculate effort.

1.4 Various Performance Measures

The evaluation of accuracy of the model can be obtained by using the following criteria:

1.4.1 Mean Square Error (MSE)

It can be calculated as:

$$MSE = \frac{\sum_{i=1}^N (y_i - \bar{y})^2}{N} \quad (1.8)$$

1.4.2 Magnitude of Relative Error (MRE)

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

$$MRE_i = \frac{|ActualEffort_i - PredictedEffort_i|}{ActualEffort_i} \quad (1.9)$$

1.4.3 Mean Magnitude of Relative Error (MMRE)

The **Mean Magnitude of Relative Error (MMRE)** can be obtained by taking the average of MRE over N observations.

$$MMRE = \frac{\sum_{i=1}^N MRE_i}{N} \quad (1.10)$$

1.4.4 Root Mean Square Error (RMSE)

It is just the square root of the mean square error.

$$RMSE = \sqrt{\frac{\sum_{i=1}^N (y_i - \bar{y})^2}{N}} \quad (1.11)$$

1.4.5 Normalized Root Mean Square(NRMS)

The **Normalized Root Mean Square(NRMS)** can be obtained by dividing the RMSE value with standard deviation of the actual effort value.

$$NRMS = \frac{RMSE}{std(Y)} \quad (1.12)$$

where Y is the actual effort for testing data set.

1.4.6 Prediction Accuracy (PRED)

The **Prediction Accuracy (PRED)** can be calculated as:

$$PRED = (1 - (\frac{\sum_{i=1}^N |actual_i - predicted_i|}{N})) * 100 \quad (1.13)$$

where

N = Total number of data in the test set.

1.5 Dataset used for Effort Calculation

The dataset from forty Java frameworks is inferred throughout two progressive semesters of graduate courses on Software Engineering. Effectiveness of the *Class Point* approach [5] is proved by the use of such data in the validation process. It is clear that the utilization of understudy's undertakings may undermine the outside legitimacy of the investigation and consequently, for the appraisal of the system; further dissection is required by utilizing information hailing from the modern world. By and by, we have attempted to make the approval process as correct as could reasonably be expected.

The proposed approach for use case point is based on one forty nine data set used in [7]. The use of this data set designates to evaluate software development effort and validate the practicability of improvement. The utilization of such information in the approval procedure has given introductory exploratory confirmation of the adequacy of the UCP.

1.6 Thesis Organization

The rest of the thesis is organized as follows.

Literature Survey is given in **Chapter-2**. Details of the Stochastic Gradient Boosting Technique) technique is given in **Chapter-3**, in that chapter proposed approach and experimental details are also described. In **Chapter-4** Results and Comparison are discussed. Discussion of conclusion and Future work are at the end **Chapter-5**.

Chapter 2

Literature Survey

Gennaro Costagliola et al. [5] purposed the class point approach method and observed that the prediction accuracy of CP1 and CP2 under the class point approach were 75 and 83 percentage respectively. They found this result by conducting an experiment on a dataset with forty projects. Wei Zhou and Qiang Liu [3] extended the above approach by adding CP3 measure in CPA. They acknowledged twenty-four system characteristics rather than the eighteen characteristics by Gennaro Costagliola et al. They observed no effect on the prediction accuracies of CP1 and CP2, although the no. of characteristics changed by this approach. S.Kanmani et al. [8] applied an artificial neural network model on CPA for mapping CP1 and CP2 into the estimated software development effort and observed that the prediction accuracy for CP1 was improved to 83 and CP2 to 87 percentage. SangEun Kim et al. [9] modified definitions of class point and used a some extra parameters beside NSR, NEM and NOA to compute the no. of class points. S. Kanmani et al. [10] applied a fuzzy system by adopting the subtractive clustering technique in the class point approach for effort calculation and comparison it with the result obtained from an artificial neural network. They observed that their technique yields better results than ANN. Mahmoud O. Elish [11] applied a data mining technique, multiple additive regression trees (MART) that augments and progresses the classification and regression trees (CART) model. The obtained results were then compared with linear regression, support vector regression and radial basis function networks models with the help of a NASA software project dataset and found an improved estimation accuracy. J. S. Pahariya et al. [12] proposed a new genetic

programming-based feature selection algorithm in software effort estimation and compared the predicted effort with other computational intelligence techniques. The results show that the new architecture for genetic programming surpassed the other techniques. Ali B. Nassif et al. [7] applied Treeboost model for estimation of software effort based on the use case point approach by considering eighty-four data points and obtained promising results.

Adriano L.I. Oliveira [13] describes a relative study on radial basis function neural networks (RBFNs), support vector regression (SVR) and linear regression for software project effort estimation. The experiment is carried out using NASA project datasets and the result shows that SVR performs better than RBFN and linear regression. K. Vinay Kumar, et al. [14] have proposed the use of wavelet neural network (WNN) to predict the software development effort and compares the result with other techniques such as radial basis function network (RBFN), multiple linear regression (MLR), multilayer perceptron (MLP), support vector machine (SVM) and dynamic evolving neuro-fuzzy inference system (DENFIS).

A. Issha et al. [15] reports on three use case model based software effort estimation models such as use case patterns estimation method, object points extraction estimation method and use case rough estimation method. The prediction accuracy of this methods have been verified using a wide spectrum of software projects. Ali B. Nassif et al., [16] presents a log-linear regression model as well as a multilayer perception model based on the use case point model (UCP) for software effort estimation. By the comparative analysis of the result, they show that the MLP model can overrun the regression model in case of small projects, but the log-linear regression model provides better prediction accuracy in case of larger projects.

Using the use case point size metric, Ali B. Nassif et al. [17] presented a regression model for software effort estimation. They proposed an effort equation that consider the non-linear relationship between software size and software effort, along with the influences of project productivity and complexity. Results intends that the software effort estimation accuracy can be improved by 16.5%. Ali B.

Nassif et al., [18] extended this process by applying mamdani fuzzy inference system with regression model to enhance the estimation accuracy and found 10% improvement in the result. Ali B. Nassif et al. [19] also applied sugeno fuzzy inference system with regression model to enhance the estimation accuracy and found 11% improvement in the MMRE result. Ali B. Nassif et al., [20] propose an Artificial Neural Network (ANN) for software effort estimation based on the Use Case Point (UCP) model with the help of 240 data points and found a competitive result with respect to other regression model. Ali B. Nassif et al., [21] also present some techniques using fuzzy logic and neural networks for improvement of the prediction accuracy of the use case points method and obtained a hike up to 22%. Bilge Bakele et al., [22] proposed a machine learning based methods and evaluate the model on public data sets, gathered from software industries. From analysis, it is found out that the usage of any one model cannot produce the optimum results for software effort estimation.

Shashank et al. [23] used Multi-Layer Perceptron (ANN) and Radial Basis Function Network (RBFN) in class point approach for software effort estimation. Then in his paper [2], he applied Adaptive Neuro-Fuzzy Inference System Model for optimization of class point parameters and got improved results. He also applied Various Support Vector Regression Kernel Methods in Use Case Point Approach for Software Effort Estimation in his article [24].

Chapter 3

Stochastic Gradient Boosting Technique

3.1 Introduction

The effort involved in software developing process plays an important role in determining the faith of the product. In the context of software development using object-oriented methodologies, in this chapter, using both class point and use case point approach the main aim is software project's effort estimation and optimize the parameters using Stochastic Gradient Boosting Technique. The results obtained from the SGB technique is compared with various types of adaptive regression techniques such as Multi-Layer Perception (ANN), Radial Basis Function Network(RBFN). By the correct estimation of the software product, we can have software with satisfactory quality inside our plan and on arranged timetables.

3.2 Stochastic Gradient Boosting Technique

The Stochastic Gradient Boosting (SGB) technique is also called the Tree-boost model [25]. Boosting means that we apply the function iteratively in a series and combine the output of each function with a weighting coefficient in order to minimize the total error of prediction and increase the accuracy. The mathematical representation of the SGB algorithm can be written as

$$F(y) = F_0 + C_1 * T_1(y) + C_2 * T_2(y) + \dots + C_M * T_M(y) \quad (3.1)$$

where $F(y)$ is the estimated target value. F_0 is the initial value for the series. Vector y is used to represent the pseudo-residual values remaining at this point in the series. To fit the pseudo-residuals, a series of trees $T_1(y)$, $T_2(y)$ etc. are used. C_1 , C_2 etc. are coefficients of the tree node estimated values that are calculated using the SGB technique.

Usually, an individual tree consists of 8 terminal nodes with depth level 3. Hence, it is fairly small. But, the full SGB model is built with hundreds of these small trees. Beginning with the first tree, successive trees are fitted to the data. The residuals (error values) from the preceding tree are fed into the next tree in order to reduce the error. After repeating the process for a chain of trees, the final predicted value is obtained by the summation of the individual weighted contributions of the trees. The SGB method uses the Huber-M loss function for regression. Residuals falling under the Hubers Quantile-Cutoff are squared before use. In other cases the absolute values are used.

Stochastic means that a random percentage of training data points (50% of all). To delay the learning process and elongate the length of the series, a shrinkage factor (between 0 and 1) is multiplied to each tree in the series. In return the increased length compensates for the shrinkage. This improves the prediction values. An Influence Trimming Factor is applied to optimize the process, as it allows the rows with small residuals to be excluded.

3.3 Proposed Approach

The proposed model is based on a dataset [5] which is derived from forty student projects built in Java programming language and aims to evaluate software-development effort. For software effort estimation of a given software project, basically the below steps have been used.

Steps in Effort Estimation

1. **Calculation of Class Points or Use case points:** After collecting the data from other developed projects, the CP2 value is computed from the class diagram. This generated CP2 value is used as an input argument. In

case of use case point the use case point (UCP) has been computed from the use case diagram.

2. **Normalization Dataset:** Input arguments are normalized over the range [0,1]. For X be the dataset and x is an element of the dataset, the normalization of the x can be expressed as :

$$\text{Normalized}(x) = \frac{x - \min(X)}{\max(X) - \min(X)} \quad (3.2)$$

where

$\min(X)$ = the minimum value of X .

$\max(X)$ = the maximum value of X .

if $\max(X)$ is equal to $\min(X)$, then $\text{Normalized}(x)$ is set to 0.5.

3. **Division of data set:** The dataset is divided into three parts. These are learning, validation and test set.
4. **Selection of Parameters:** Various parameters such as number of trees, Hubers quantile cutoff, shrinkage factor, stochastic factor and influence trimming factor are selected to perform model selection using a five fold cross-validation procedure.
5. **Performing Model Selection:** In this step, a five-fold cross validation is implemented for model selection. The model that provides the lowest normalized root mean square error (NRMSE), mean magnitude of relative error (MMRE) values and the highest prediction accuracy (PRED) values is selected as the best model for each fold.
6. **Performance Evaluation:** The performance of the model is verified using final NRMSE, MMRE and PRED values obtained from test samples. These final values are calculated by taking the average of NRMSE, MMRE and PRED accuracy values obtained in each fold.

The above steps are followed to implement the SGB-based effort estimation model. Finally, a comparison of the results obtained using the SGB-based effort estima-

tion model with the results obtained from the MLP and RBFN-based models is presented to assess their performances.

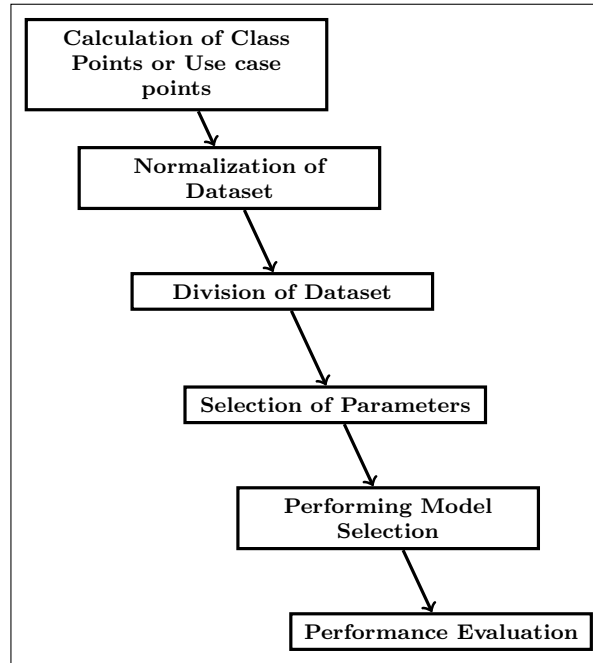


Figure 3.1: Proposed Steps Used for Effort Estimation using SGB

3.4 Experimental Details

After computing value of the final class point and use case point values, the dataset is then normalized. The normalized dataset is partitioned into distinctive subsets using a double sampling method. In starting step, the normalized dataset is partitioned into a training and a test set. The training set helps in learning purposes (model estimation), whereas the test set help in the prediction accuracy estimation of the final model. In the second step, the training set is partitioned into a validation set and a learning set. The learning set helps in model parameter’s estimation, and the validation set helps in selection of an model with optimal performance measures (usually via cross-validation).

Every fifth row of the table, is extracted for testing purposes and the rest are used for training purposes. Hence, after completion of the first step of the double sampling process, the complete forty project dataset is divided into thirty-two rows for training and eight rows for testing. Then every fifth row of the training set is

selected for validation purposes, and the rest are used for learning purposes. So, after completion of the second step of the double sampling process, the complete thirty-two project dataset is partitioned into twenty-six rows for learning and six rows for validation. After partitioning the sample into a learning set and a validation set, using a five fold cross-validation process the model selection is performed.

3.5 Model Design Using Stochastic Gradient Boosting Technique

To design an effort estimation model using the SGB technique, the following steps are used.

1. The coefficient of F0 is obtained by calculating the mean of the target variables (Software Effort).
2. A random percentage of rows are selected to feed the next tree using the stochastic factor. If it is set to 0.5, 50 percent of the rows will be randomly chosen.
3. The residuals of the rows are sorted and then the residues using the Hubers Quantile Cutoff factor are transferred. The transformed residual values are called pseudo-residuals.
4. The first tree (T1) is fitted to the pseudo-residuals.
5. The predicted values of the nodes are calculated using the mean of the pseudo-residuals in each of the terminal nodes.
6. The residuals between the predicted values and the pseudo-residuals that fed the tree are calculated.
7. The Hubers Quantile Cutoff factor is applied again on the result obtained from step 6 and the mean of these residuals are then computed.

8. The boost coefficient (A1) of the tree is obtained by measuring the difference between the mean residual value and the mean of the predicted values of the tree.
9. Finally, the boost coefficient is multiplied by the shrink value to retard the learning process.

The following parameter values are used to predict the effort using the SGB technique.

- No of Trees : 1000
- Hubers Quantile Cut off : 0.95
- Shrinkage Factor : 0.1
- Stochastic Factor : 0.5
- Influence Trimming Factor : 0.01

3.6 Summary

In this chapter, Stochastic Gradient Boosting Technique has been proposed and details of steps are described to predict the effort of software product using optimized class point and use case point. First, the calculated class point and use case point values are being normalized into a range of values and used as input arguments to the SGB model to optimize the effort estimation performance measures. The optimization will be achieved by implementing Stochastic Gradient Boosting technique. Finally, the produced results from distinctive systems have been looked at for assessing the execution of diverse models.

Chapter 4

Results and Comparison

4.1 Class Point Approach Result

To improve reliability of software development processes proper estimation of effort is an essential part. Among various cost estimation methods in this chapter, the estimation of the effort of various software projects is done using Class Point Approach. The parameters are optimized using the Stochastic Gradient Boosting technique and compare with the MLP and RBFN techniques to achieve better accuracy.

Table 4.1: MMRE and NRMSE Values Obtained using SGB for Each Fold

Fold	MMRE	Test Set Prediction Error (NRMSE)
1	0.3920	0.3130
2	0.1684	0.4819
3	0.5199	0.3736
4	0.0790	0.1582
5	1.0079	0.2460
Average	0.4334	0.3145

Table 4.1 provides the minimum MMRE and NRMSE values obtained from the training set and the test set using the SGB technique for each fold in the five fold cross validation processes. The final result is calculated by taking the average of the MMRE and NRMSE values obtained in each fold. The proposed model generated using the SGB technique is plotted based on the training and testing samples in Figure 4.1.

From Figure 4.1, it is observed that the training and testing data are correlated

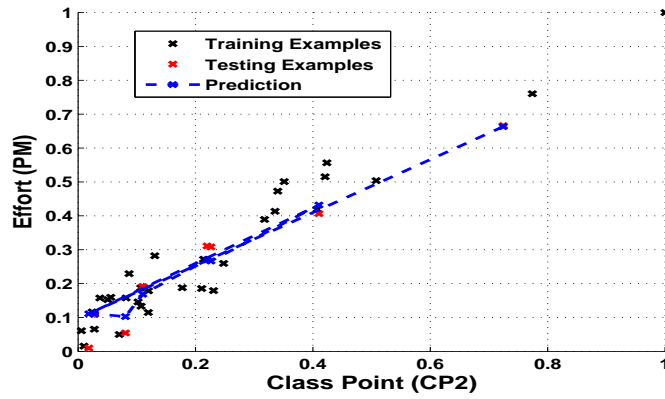


Figure 4.1: The SGB-based Effort Estimation Model for class point approach

around the regression line.

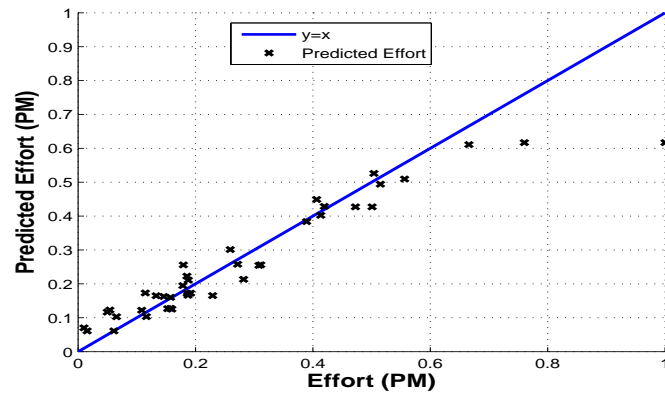


Figure 4.2: Actual vs. Predicted Effort using SGB for class point approach

Figure 4.2 displays the actual effort and the predicted effort obtained using the SGB technique. From this figure, it is observed that there is very little difference between the predicted effort and the actual effort.

4.1.1 Comparison

On the basis of the results obtained, the estimated effort values using the MLP, RBFN and SGB techniques are compared. The results for the MLP and RBFN techniques are collected from Satapathy et al. [23]. The results show that the SGB technique performs better than the MLP and RBFN techniques.

Table 4.2 compares the actual effort with an estimated effort generated using the MLP, RBFN and SGB techniques. From the table, it is observed that the

Table 4.2: Comparison of Efforts obtained using MLP, RBFN and SGB

	Actual Effort	MLP Effort	RBFN Effort	SGB Effort
1	0.6661	0.7218	0.7700	0.6111
2	0.0100	0.0812	0.0839	0.0703
3	0.4067	0.5135	0.4689	0.4490
4	0.3081	0.2328	0.2822	0.2545
5	0.3112	0.2262	0.2757	0.2564
6	0.1916	0.1742	0.1687	0.1725
7	0.0542	0.1553	0.1410	0.1228
8	0.1084	0.0948	0.0924	0.1223

difference between the predicted effort and the actual effort values for the SGB technique is much less than for the MLP and RBFN techniques. Hence, it can be concluded that the SGB technique exhibits higher prediction accuracy than the MLP and RBFN techniques. The results obtained from table Table 4.1

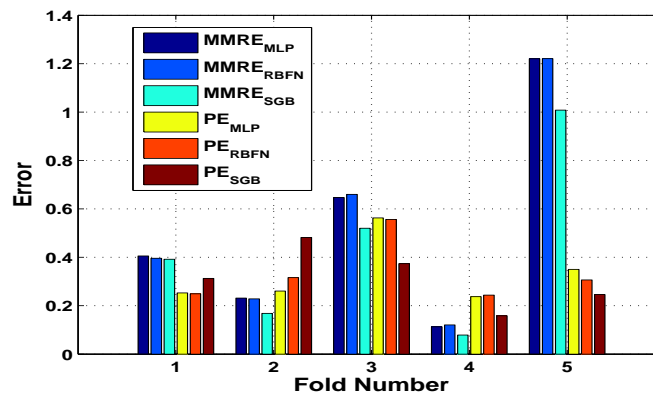


Figure 4.3: Comparison of MMRE and PE Values

are plotted in figure 4.3. This figure shows a comparison between MMRE and the prediction errors obtained using the MLP, RBFN and SGB techniques. In this figure, $MMRE_{MLP}$, $MMRE_{RBFN}$ and $MMRE_{SGB}$ denote the MMRE values obtained using these three techniques. Similarly, PE_{MLP} , PE_{RBFN} and PE_{SGB} denote the prediction errors obtained using these three techniques.

Table 4.3 provides a comparison of the results obtained by relevant articles as mentioned in the related work section. The performance of the techniques used in those articles is compared by measuring their prediction accuracy (PRED) values. The results show that the first two articles provide the same prediction accuracy;

Table 4.3: 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 [3]	Regression Analysis	83%
3	S. Kanmani et al. [8]	Neural Network	87%
4	S. Kanmani et al. [10]	Fuzzy Logic	92%

whereas the third and fourth articles show significant improvement in the prediction accuracy. Finally, the results obtained in the related work section are compared with the results of the proposed approaches, which are shown in Table 4.4.

Table 4.4: Comparison of Efforts obtained using MLP, RBFN and SGB

Sl. No.	Techniques	MMRE	NRMSE	Prediction Accuracy (in percentage)
1	Multi-Layer Perception	0.5233	0.3327	94.8185
2	Radial Basis Function Network	0.5252	0.3344	94.7539
3	Stochastic Gradient Boosting	0.4334	0.3145	95.3261

Better results in evaluation are indicated by lower values of MMRE, NRMSE and higher values of prediction accuracy. Table 4.4 displays the final comparison of MMRE, NRMSE and the prediction accuracy values for the MLP, RBFN and SGB techniques.

4.2 Use Case Point Approach Result

The proposed model generated using the SGB technique for UCP have been plotted based on the training and testing sample data set as shown in Figure 4.4. The graphs show the variation of predicted effort value with respect to its corresponding use case point value. In these graphs, it is clearly shown that the data points are very little dispersed than the regression line. Hence the correlation is higher. While comparing the dispersion of data points from the predicted model in the above graphs for UCP, it is clearly visible that for SGB based UCP model, the data points are less dispersed. Hence, the models exhibit less error values and higher prediction accuracy values.

From Figure 4.4, it is observed that the training and testing data are correlated

around the regression line. Figure 4.5 displays the actual effort and the predicted

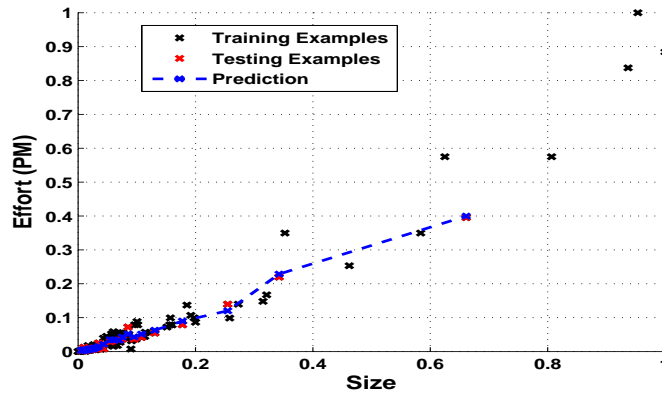


Figure 4.4: Actual vs. Predicted Effort using SGB for use case point

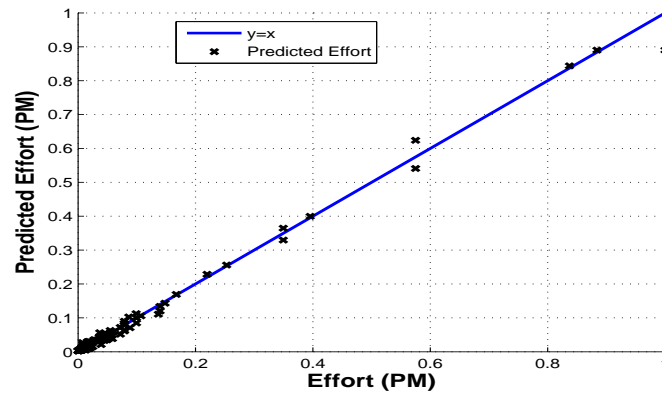


Figure 4.5: The SGB-based Effort Estimation Model for use case point

effort obtained using the SGB technique. From this figure, it is observed that there is very little difference between the predicted effort and the actual effort.

4.2.1 comparison

On the basis of the results obtained, the estimated effort values using the MLP, RBFN, Log Linear regression and SGB techniques are compared. The results for the MLP, RBFN and Log Linear regression techniques are collected from Ali B. Nasif et al. [20] and [17]. We collected results for SVR kernel methods for UCP from shashank [24]. The results show that the SGB technique gives better results than the other techniques.

Table 4.5: Comparison of Prediction Accuracy Values of Related Works

Sl. No.	Related Papers	Technique Used	Prediction Accuracy
1	A. Issha et al. [15]	3 Novel UCP model	67%
2	Ali B. Nasif et al. [20]	ANN Model	90.27%
3	Ali B. Nasif et al. [17]	Regression Model	95.8%

Table 4.5 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 maximum of 95% prediction accuracy is achieved using regression analysis technique for UCP. Finally, the results obtained in related work section is compared with results of proposed approaches, which is shown in Table 4.6. The results obtained using proposed technique shows improvement in the prediction accuracy value.

Table 4.6: Comparison of MMRE, NRMSE and PRED Values

	MMRE	Prediction Accuracy (in percentage)
Multi-Layer Perceptron	0.5233	94.8185
Log Linear Regression	0.5252	94.7539
SVR Linear Kernel	0.5438	97.7421
SVR Polynomial Kernel	1.0003	97.4089
SVR RBF Kernel	0.3857	98.0188
SVR Sigmoid Kernel	0.6049	97.3931
Stochastic Gradient Boosting	0.2845	98.3635

When the MMRE and prediction accuracy is utilized as a part of assessment, better results are shown by lower values of MMRE and higher values of prediction accuracy. Table 4.6 displays the last comparison of MMRE, NRMSE and

prediction accuracy values for the MLP, RBFN and SGB techniques.

4.3 Summary

In this chapter, to estimate the effort of an object-oriented system, models using the Stochastic Gradient Boosting Technique has been proposed; and their results have been compared. The output shows that Stochastic Gradient Boosting Technique using class point and use case point give better results.

Chapter 5

Conclusion and Future Work

A number of approaches have been considered by researchers and practitioners for the effort estimation required to develop a given software product. The class point model and the use case point are example of the effort estimation models and they are preferred for projects developed using object-oriented technologies. This model also helps in calculating the effort during an initial stage of the software development life cycle. In the described paper, the class point model and the use case point are enhanced using the SGB technique and the generated results are compared with the results obtained from the MLP, RBFN and SVR techniques. The results show that the SGB-based effort estimation model possesses lower MMRE, NRMSE and higher prediction accuracy. So, a conclusion can be derived that the effort estimation using the SGB-based model provides results with more accuracy than the MLP, SVR and RBFN-based models. The computations for the procedure were implemented and the outputs were generated using MATLAB. The research work carried out in this paper can also be extended by applying other machine learning techniques such as Decision Tree Forest, Random Forest etc. on the class point and use case point approaches and comparing their results with the results of the SGB technique to measure their accuracies.

Bibliography

- [1] J. Matson, B. Barrett, and J. Mellichamp, “Software development cost estimation using function points,” *Software Engineering, IEEE Transactions on*, vol. 20, no. 4, pp. 275–287, 1994.
- [2] S. M. Satapathy, M. Kumar, and S. K. Rath, “Fuzzy-class point approach for software effort estimation using various adaptive regression methods,” *CSI Transactions on ICT*, pp. 1–14, 2013.
- [3] W. Zhou and Q. Liu, “Extended class point approach of size estimation for oo product,” in *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, vol. 4, pp. 117–122, IEEE, 2010.
- [4] G. Costagliola, F. Ferrucci, G. Tortora, and G. Vitiello, “Towards a software size metrics for object-oriented systems,” *Proc. AQUIS*, vol. 98, pp. 121–126, 1998.
- [5] 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*, vol. 31, no. 1, pp. 52–74, 2005.
- [6] G. Karner, “Resource estimation for objectory projects,” *Objective Systems SF AB*, vol. 17, 1993.
- [7] A. B. Nassif, L. F. Capretz, D. Ho, and M. Azzeh, “A treeboost model for software effort estimation based on use case points,” in *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, vol. 2, pp. 314–319, IEEE, 2012.

-
- [8] S. Kanmani, J. Kathiravan, S. S. Kumar, and M. Shanmugam, “Neural network based effort estimation using class points for oo systems,” in *Proceedings of the International Conference on Computing: Theory and Applications, ICTA '07*, (Washington, DC, USA), pp. 261–266, IEEE Computer Society, 2007.
- [9] 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*, pp. 415–421, 2006.
- [10] S. Kanmani, J. Kathiravan, S. S. Kumar, and M. Shanmugam, “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*, (New York, NY, USA), pp. 141–142, ACM, 2008.
- [11] M. O. Elish, “Improved estimation of software project effort using multiple additive regression trees,” *Expert Systems with Applications*, vol. 36, no. 7, pp. 10774–10778, 2009.
- [12] J. Pahariya, V. Ravi, and M. Carr, “Software cost estimation using computational intelligence techniques,” in *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pp. 849–854, IEEE, 2009.
- [13] A. L. Oliveira, “Estimation of software project effort with support vector regression,” *Neurocomputing*, vol. 69, no. 13, pp. 1749–1753, 2006.
- [14] K. Vinay Kumar, V. Ravi, M. Carr, and N. Raj Kiran, “Software development cost estimation using wavelet neural networks,” *Journal of Systems and Software*, vol. 81, no. 11, pp. 1853–1867, 2008.
- [15] A. Issa, M. Odeh, and D. Coward, “Software cost estimation using use-case models: A critical evaluation,” in *Information and Communication Technologies, 2006. ICTTA '06. 2nd*, vol. 2, pp. 2766–2771, IEEE, 2006.

-
- [16] A. B. Nassif, D. Ho, and L. F. Capretz, "Towards an early software estimation using log-linear regression and a multilayer perceptron model," *Journal of Systems and Software*, 2012.
- [17] A. B. Nassif, D. Ho, and L. F. Capretz, "Regression model for software effort estimation based on the use case point method," in *2011 International Conference on Computer and Software Modeling*, pp. 117–121, 2011.
- [18] A. B. Nassif, L. F. Capretz, and D. Ho, "A regression model with mamdani fuzzy inference system for early software effort estimation based on use case diagrams," in *Third International Conference on Intelligent Computing and Intelligent Systems*, pp. 615–620, 2011.
- [19] A. B. Nassif, L. F. Capretz, and D. Ho, "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*, pp. 393–398, IEEE, 2011.
- [20] A. B. Nassif, L. F. Capretz, and D. Ho, "Estimating software effort using an ann model based on use case points," in *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, vol. 2, pp. 42–47, IEEE, 2012.
- [21] A. B. Nassif, "Enhancing use case points estimation method using soft computing techniques," *Journal of Global Research in Computer Science*, vol. 1, no. 4, 2010.
- [22] B. Baskeles, B. Turhan, and A. Bener, "Software effort estimation using machine learning methods," in *Computer and information sciences, 2007. iscis 2007. 22nd international symposium on*, pp. 1–6, IEEE, 2007.
- [23] S. M. Satapathy, M. Kumar, and S. K. Rath, "Class point approach for software effort estimation using soft computing techniques," in *Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on*, pp. 178–183, IEEE, 2013.

- [24] S. M. Satapathy and S. K. Rath, “Use case point approach based software effort estimation using various support vector regression kernel methods,” *arXiv preprint arXiv:1401.3069*, 2014.
- [25] J. H. Friedman, “Stochastic gradient boosting,” *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.

Dissemination of Work

Accepted

1. Shashank Mouli Satapathy, Barada prasanna Acharya, Santanu Kumar Rath. Class Point Approach for Software Effort Estimation using Stochastic Gradient Boosting Technique. *ACM SIGSOFT Software Engineering Notes*, ACM, May 2014.