**Optimizing Software Effort Estimation Accuracy**

**with a Machine Learning Model**

**A PROJECT REPORT**

***Submitted by***

**SANTHAGAROOBAN D**

**(201021101)**

***In partial fulfilment for the award of the degree of***

**BACHELOR OF ENGINEERING IN COMPUTER SCIENCE**

**ENGINEERING**

** **

**IFET COLLEGE OF ENGINEERING**

(An Autonomous Institution)

*Approved by AICTE, New Delhi and Accredited by NAAC & NBA*

*Affiliated to Anna University, Chennai-25*

Gangarampalayam, Villupuram – 605 108

**NOVEMBER 2023**

**IFET COLLEGE OF ENGINEERING**

(An Autonomous Institution)

**BONAFIDE CERTIFICATE**

Certified that this project report **“Optimizing Software Effort Estimation Accuracy** **with a Machine Learning Model”** is the bonafide work of **“SANTHAGAROOBAN.D[201021101]”** who carried out the mini project work under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**SIGNATURE SIGNATURE**

**Dr. P. KANIMOZHI, Ph.D Mr.D.RAGHU RAMAN , M.E**

HEAD OF THE DEPARTMENT, SUPERVISOR,

Professor, ASSOCIATE PROFESSOR,

Department of CSE, Department of CSE,

IFET College of Engineering, IFET College of Engineering,

Villupuram. Villupuram.

The project report submitted for the viva voce held on………….

Signature of the Internal Examiner Signature of the External Examiner

**ACKNOWLEDGEMENT**

I thank the almighty, for the blessings that have been showered upon me to bring forth the success of the project. I would like to express my sincere gratitude to our Chairman **Mr.K.V.RAJA**, and our Secretary **Mr.K.SHIVARAM ALVA** for providing an excellent infrastructure and necessary resources to carry out this project and I extend my gratitude to our principal **Dr.G.MAHENDRAN**, for his constant support to my work.

I also take this opportunity to express my sincere thanks to our Vice Principal and Dean Academics **Dr.S.MATILDA**, and our Head Placement and Corporate Affairs **Prof.S.VISWANATHAN**, who has provided all the needful help in executing the project successfully.

I wish to express my thanks to our Head of the Department

**Prof.P.KANIMOZHI**, Professor for her persistent encouragement and support to complete this project. I express my heartfelt gratitude to my guide **Mr. D.RAGHU RAMAN**, ASSOCIATE Professor, Department of Computer Science and Engineering for her priceless guidance and motivation which helped me to bring this project to a perfect shape.

I express my deep sense of thanks to all faculty members and lab technicians in my department for their cooperation and interest shown at every stage of our endeavour in making a project work success.

Last but not the least, I whole heartedly thanks to our lovely parents and friends for the moral support in tough times and their constructive criticism which made me to succeed in my work.

**ABSTRACT**

In the dynamic realm of software development, accurately estimating effort is crucial for successful project completion within the constraints of time, budget, and resources. Traditional effort estimation methods often fall short due to their subjective nature and inability to capture the complex interplay of factors that influence development effort. Machine learning (ML) offers a promising avenue for addressing these limitations and improving estimation accuracy. By leveraging the power of algorithms to learn from historical data, ML models can identify patterns and relationships between project characteristics and actual effort, leading to more informed and reliable estimates. This project aims to develop and optimize an ML model for software effort estimation, considering factors such as project size, complexity, team size, and module dependencies. The model will be trained on a comprehensive dataset of historical software projects, utilizing various ML techniques to capture the nuances of effort estimation. The optimized model will be evaluated on its ability to predict effort accurately for new projects, with a focus on improving the accuracy of time, budget, member, and module estimations. The potential benefits of this project include enhanced project planning, resource allocation, risk management, and overall project success.

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
|  | **ABSTRACT** |  |
|  | **LIST OF FIGURES** |  |
|  | **LIST OF ABBREVIATION** |  |
| **1** | **INTRODUCTION** |  |
|  | 1.1 Motivation |  |
|  | 1.2 Problem Statement |  |
|  | 1.3 Research Objective |  |
|  | 1.4 Machine Learning Techniques Used |  |
|  | 1.4.1 Decision Tree Technique  1.4.2 Stochastic Gradient Boosting  Technique  1.4.3 Random Forest Technique  1.4.4 Support Vector Regression Technique |  |
| **2** | **LITERATURE SURVEY** |  |
|  | 2.1 Survey on Basic Software Estimation Techiniques |  |
|  | 2.2 Survey on Class Point Approach |  |
|  | 2.3 Survey on Use Case Point Approach |  |
|  | 2.4 Survey on Effort Estimation of Web Application |  |
|  | 2.5 Survey on Story Point Approach for Agile Software Effort Estimation |  |
|  | 2.6 Survey on Software Effort Estimation using Machine Learning Techniques |  |
|  | 2.7 Summary of Observations |  |
|  | 2.8 Existing System |  |
| **3** | **CLASS POINT APPROACH FOR SOFTWARE EFFORT ESTIMATION USING MACHINE LEARNING TECHNIQUES** |  |
|  | 3.1 Introduction |  |
|  | 3.2 Methodologies Used |  |
|  | 3.2.1 Class Point Approach |  |
| **4** | **PROPOSED SYSTEM** |  |
|  | 4.1 Experimental Details |  |
|  | 4.1.1 Model Design using Stochastic  Gradient Boosting Technique |  |
|  | 4.1.2 Model Design using Various SVR  Kernel Methods |  |
| **5** | **USE CASE POINT APPROACH FOR SOFTWARE EFFORT ESTIMATION USING MACHINE LEARNING TECHNIQUES** |  |
|  | 5.1 Introduction |  |
|  | 5.2 Methodologies Used |  |
|  | 5.2.1 Use Case Point Approach |  |
| **6** | **CONCLUTION** |  |
|  |  |  |

# CHAPTER 1

# INTRODUCTION

Estimation of eﬀort is considered to be a primary activity under the broad aspects of software project management, which is defined as the process of planning and controlling the development of a system at an optimal cost meeting the right set of requirements. It is an acknowledged fact that a good number of software fail due to faulty project management practices. Each year billions of dollars are wasted on entirely preventable mistakes.

The various common factors behind the failure of a software project are:

* + - Unrealistic or unarticulated project goals
    - Inaccurate estimates of needed resources
    - Badly defined system requirements
    - Poor reporting of the project’s status
    - Unmanaged risks
    - Poor communication among customers, developers, and users
    - Use of immature technology
    - Inability to handle the project’s complexity
    - Sloppy development practices
    - Poor project management
    - Stakeholder politics
    - Commercial pressures

Therefore, it is quite necessary to adhere to key aspects of software project management activities. The software project estimation is considered as the most diﬃcult and challenging task among all these features. Project estimation involves estimation of size, eﬀort, cost, time, and staﬃng. For any software development project, the size of the product is often estimated at the very beginning stage. Taking input of the size of software, the eﬀort needed are identified. From eﬀort estimation, product duration and cost are found out.

Software size estimation is an important feature in order to determine the eﬀort required to develop a software product. It is the methodology of anticipating the most practical measure of exertion (conveyed as individual hours or capital) needed to create or keep up development tasks in light of inadequate, questionable and uproarious data. Software Eﬀort Estimation (SEE) is the procedure of foreseeing the most sensible utilization of eﬀort required in order to develop or maintain software. SEE is the activity of estimating the total eﬀort required to complete a software project. Eﬀectively assessing the eﬀort required in order to develop a software product is of fundamental significance in order to sustain competitiveness in the market. Both under and over-estimation prompts undesirable results for the organizations. Under-estimation may bring about overwhelms in budget and schedule, which consequently may bring about the cancellation of projects; in this way, squandering the whole eﬀort spent until that point. Over-estimation may bring about promising projects not to be subsidized; consequently, hurting the organizational capabilities. The process of eﬀort estimation needs to be optimized because proper estimates are necessary both on the developer side as well as client side. On the developer side, estimates help in planning the development and monitoring the progress. While on the client side, they are used for negotiating contracts, setting completion dates, prototype release dates etc. However, as indicated in the research work reported by the Brazilian

Ministry of Science and Technology-MCT, just 29% of the organizations fulfilled size estimation and 45.7% achieved software eﬀort estimation. So, the research work on eﬀort estimation of proposed software has invited attention of a number of practitioners and theoreticians.

In the year 2013, the Standish Group Chaos Manifesto states that 43% of IT projects were delivered late, over budget, and/or with less than the required features and functions. This indicates that the role of project management is being increasingly accepted as a more important aspect for sustainability. The International Society of Parametric Analysis (ISPA) recognized the principle purposes behind failures of a majority of software.

These reasons can be abridged as follows:

* + - * Lack of understanding the requirement
      * Improper software size estimation
      * Lack of evaluation of the staﬀs expertise level

Another Standish report outlines diﬀerent principal factors, that expedite the failure of a software project such as:

* Realistic estimation
* Uncertainty in requirements of system and software
* Lack of skilled estimators
* Limitation in Budget
* Optimized software estimation process
* Lack of historical data
* Failed to consider historical data

In a nutshell, it is observed from the above parameters that numerous software projects fizzle due to incorrectness in software estimation process and poor understanding or inadequacy of the prerequisites. Hence, to obtain right kinds of results in estimating software eﬀort, it is essential to consider the above issues and try to resolve them as much as possible. In the present day scenario, the object-oriented concept is the accepted practice of software development. As *class* and *use case* are the basic logical unit of an object-oriented system, the use of Class Point Approach (CPA) and Use Case Point Approach (UCP) to estimate the project eﬀort help to guide the estimator in a more meaningful way. Web-based software projects are diﬀerent than conventional object-oriented projects, and hence the task of estimation for these projects is a complex one. As per Reifer, eﬀort estimation models, which are helpful for conventional software development, are not extremely precise for eﬀort estimation of web-based software development. For eﬀort estimation of web applications, the dataset of past web development projects is collected from ISBSG dataset. Similarly, in case of agile projects, Story Point Approach (SPA) is used to measure the eﬀort required to implement a user story. By adding up the estimates of user stories that were finished during an iteration (story point iteration), the project velocity is obtained. The eﬃciency of the models obtained using CPA, UCP, Web and SPA can be improved by employing certain intelligent techniques on them. The proposed research study considers the application of various machine learning (ML) techniques such as Decision Tree (DT), Stochastic Gradient Boosting (SGB), Random Forest (RF) and Support Vector Regression (SVR) kernel methods over CPA, UCP, Web and SPA datasets in order to improve their prediction accuracies. These datasets are chosen by based on their contents and its relevance in order to employ eﬀort estimation process on those datasets. The Class Point dataset are collected from, the UCP dataset are collected from 3 diﬀerent sources, which includes dataset from industries and some are available for educational research purpose. The entire web dataset is collected from ISBSG repository and the SPA dataset are collected from. The detailed description about these datasets are presented in the contributory chapters. The results of various models obtained after applying machine learning techniques are compared with each other as well as with the results available in the literature, in order to assess their performance.

### Motivation

### The motivation for this thesis is essentially to provide the estimating community with a fresh approach to the estimation problem, which might complement present practices. The main reasons for this are:

* + 1. **Unimpressive results from algorithmic models:** Numerous empirical studies have been carried out by a number of authors in literature on the accuracy of algorithmic models. But somehow, the over-riding trend is inaccuracy and inconsistency. It may be possible to explore techniques other than algorithmic models in order to build eﬀort prediction systems. One of the major problems with the use of algorithmic models is that they are dependent on quantifiable inputs. This often renders them ineﬀective during the early stages of a software project’s conception. More appropriate approaches need to be found which can make estimates using the type of data those are present during the early stages of a project.
    2. **Lack of appropriate techniques for estimation of softwares developed using object-oriented methodology:** Object-oriented methodology is an approach of software development in the present-day scenario. But function point and COCOMO are the approaches which are still popular in the industries for eﬀort estimation of object-oriented softwares. These techniques mostly depend on lines of code, which is obtained from the coding phase of software development life cycle (SDLC). Hence, for eﬀort estimation during early stage of software development, i.e., starting with requirement analysis and design phase, more concentration should be given to estimate the eﬀort of object-oriented softwares from UML diagrams.
    3. **Absence of applicable procedures for estimation of eﬀort required to develop web-based applications:** Web-based software projects being considered in the present-day scenario are diﬀerent from conventional object-oriented projects, and hence the task of estimation for this category is a complex one. Eﬀort estimation models, which are helpful for conventional software development, are not extremely precise for eﬀort estimation of web-based software development, because traditional eﬀort estimation techniques are not adequate to capture specific features of the development which can influence the size and eﬀort required in the development of web-based applications.
    4. **Unavailability of proper estimation techniques for softwares developed using agile methodologies:** Agile methodologies are gaining popularity year by year in software development industries. But due to lack of proper estimation techniques for softwares developed using agile methodology, failure rates are also more. a number of agile methodologies such as scrum, extreme programming, lean programming etc. are followed by diﬀerent industries for development of their softwares. Hence, it is quite diﬃcult to propose a single estimation technique for softwares developed using diﬀerent agile methodology.

### 1.2 Problem Statement

### It has been observed from earlier research that, almost one-third number of projects surpass their budget and are conveyed late. Two-third number of projects invade their original estimates. It is an exceptionally troublesome assignment for a manager or system analyst to anticipate with much correctness the eﬀort required to develop a software, when a number of external parameters such as unclear project definition, technological uncertainty, implementation complexity, team experience etc. play a significant role. Hence, project managers usually are not able to determine truly, how much time and manpower a successful project needs. However, to help the organization in developing qualitative products inside planned period during the early stage of SDLC, legitimate estimation of software eﬀort is essential.

### 1.3 Research Objective

### This section indicates the progress stepped towards the above discussed state-of-the-art issues. The objectives of the research work outlined in this thesis are as follows:

1. To estimate the eﬀort required to develop an object-oriented software utilizing class point approach and improve the prediction accuracy of the result using diﬀerent machine learning techniques.
2. To propose diﬀerent machine learning techniques based eﬀort estimation model for object-oriented softwares using use case point approach.
3. To assess the eﬀectiveness of applying machine learning techniques for eﬀort estimation of web-based applications and validate the result using industry dataset.
4. To analyze and compare the application of diﬀerent machine learning techniques for eﬀort estimation process of softwares developed using scrum based agile methodology.

Hence the overall research objective of this thesis is to estimate the eﬀort of a software product using Class Point (CP), Use Case Point (UCP), Web and Story Point (SP) approaches. Then optimization of various parameters has been achieved using various ML techniques to obtain better prediction accuracy. Finally, the prediction accuracy obtained using diﬀerent ML techniques have been compared in order to access their performance.

### Machine Learning Techniques Used

### The following machine learning techniques are applied over the various datasets considered to calculate the eﬀort of a software product. The decision about choosing a machine learning technique for implementation purpose in the proposed research is performed based on the past research study done in the literature survey. Many researchers are applied some of the following machine learning techniques for their research purpose earlier. But none of these techniques are applied earlier for eﬀort estimation using CP, UCP, Web and SP datasets. Every proposed contribution also describes a detailed presentation about the result obtained using these techniques for their corresponding dataset. Each contribution also depicts the in detailed comparison of these techniques with earlier result obtained from literature in order to access their performance.

#### Decision Tree Technique

#### A Decision Tree (DT) is an intelligent model characterized by a binary tree that illustrates the prediction of a dependent variable using a set of predictor variables. The primary DT model was proposed by Morgan and Sonquist in 1963 and was called *Automatic Interaction Detection (AID)*. This perspective was developed further by the THAID program in 1973. The fundamental point of interest of a DT model is that it can help a novice to investigate the master plan of a specific issue. In any case, the fundamental inconvenience of a DT model is that every node is optimized locally rather than global optimization of the entire tree. Besides, DT models may experience the ill eﬀects of the over-fitting issue, and in addition from giving good accuracy in contrast with diﬀerent models.

#### Stochastic Gradient Boosting Technique

#### The Stochastic Gradient Boosting (SGB) technique is also called as the Tree-boost model . “Boosting” 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. 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*) + *....* + *CM* × *TM* (*y*)

(1.1)

where *F* (*y*) is the estimated target value and *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.

Often it is observed that, an individual tree consists of eight terminal nodes with depth level 3. Hence, it is fairly small. But, the full SGB model is built with large numbers 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 weighted contributions of individual trees. The Tree-boost method uses the *Huber-M loss function* for regression. Residuals falling under the *Huber’s Quantile-Cutoﬀ* are squared before use. In other cases the absolute values are used.

Literally *“Stochastic”* means a random percentage of training data points i.e., 50% is recommended, are used for each iteration instead of all. In order 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 activity 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.

* + 1. **Random Forest Technique**

Random Forest (RF) is an ensemble learning technique used for classification and regression purposes. 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 which are competitive 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 the RF is that it generates a number of classification trees with the help of a random vector ‘*λ*’ and an input vector ‘*x*’. A random vector ‘*λk*’ is produced for the *k*th tree, which is autonomous of the previous random vectors *λ*1*, ..., λk*−1 with equal distribution. A tree is developed using the training set and *λk*, which generates a classifier h(x, *λk*), where ‘*x*’ is an input vector. To categorize new object from an input vector, the input vector ‘*x*’ is jotted down along with 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 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 predictions for individual tree.

RF for regression purpose are created by developing trees relying upon a random vector *λ*, which is specified as the tree predictor h(x, *λ*) that 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

*Ex,Y* (*Y* − *h*(*x*))2

(1.2)

By calculating the average value obtained over *k* trees h(x, *λk*); the RF predictor is modeled.

#### Support Vector Regression Technique

#### Support Vector Machines (SVM) are a category of learning machines, helpful for implementing the structural risk minimization inductive principle in order to obtain a good generalization on a limited number of learning patterns. A version of SVM for regression was proposed by Vapnik et al in 1996. This method is called as *support vector regression (SVR)*. It is very often observed that any neural networks suﬀers from two major drawbacks. First of all, neural networks often converge on local minima rather than global minima. Secondly, neural networks often over-fit which means, if training on a pattern goes on too long, then it may consider noise as part of pattern. SVR technique does not suﬀer from either of these two drawbacks and have the advantages due to which it can be successfully used for regression task. Firstly it has a regularization parameter, which makes the user consider staying away from over-fitting. Further more it utilizes the kernel trick, so that expert knowledge regarding the issues can be build through optimizing the kernel. Thirdly a SVR is characterized by a convex optimization issue. Ultimately, it is an estimate to a bound on the test error rate, and there is a significant assemblage of hypothesis behind it, which proposes it ought to be a smart thought.

Suppose, for a given training data (*x*1, *y*1), . . . , (*xl*, *yl*), where *x* ∈ R*n* denotes the space of the input patterns and *y* ∈ R denotes its corresponding target value, the goal of regression may be identified as to find the function *f* (*x*) that best models the training data. For the case of nonlinear regression, *f* (*x*) = ⟨*w, ϕ*(*x*)⟩ + *b*, where *ϕ* is a nonlinear function which maps the input space to a higher (maybe infinite) dimensional feature space and ⟨*.,.*⟩significant.

**CHAPTER 2**

**LITERATURE SURVEY**

Software Eﬀort Estimation (SEE) is one of the important activities carried out before going ahead with development activities of proposed software. To deal with challenges in estimation of proposed software, various researchers and practitioners have proposed diﬀerent approaches. This chapter presents a survey of various approaches for software eﬀort estimation. The chapter has been divided into various sections. The section 2.1 presents the survey of various techniques proposed for basic software eﬀort estimation. These include popular techniques such as algorithmic models i.e., SLIM, Function Point, COCOMO etc., expert judgment and estimation by analogy. Section 2.2 deals with presenting various articles related to class point approach based software eﬀort estimation procedure. Section 2.3 presents the survey of articles dealing with use case point approach based software eﬀort estimation. Section 2.4 surveys articles deal with eﬀort estimation of web application. Similarly, section 2.5 presents articles providing procedures for agile software eﬀort estimation. Finally, section 2.6 presents the survey of various articles focusing on various machine learning techniques for software eﬀort estimation procedure.

**2.1 Survey on Basic Software Eﬀort Estimation Techniques**

The Software Life-cycle Management (SLIM) model, which is otherwise called Putnam model was proposed by Lawrence Putnam in 1978. The SLIM depicts the eﬀort and time required to complete the development of software of a specific size. The time-eﬀort curve of Putnam model follows the Rayleigh distribution. Function Points measure the functionality of a software as opposed to SLOC, which measures the physical component of a software. It was developed by Allan Albrecht in 1979. The International Function Point Users Group (IFPUG) defines the standard procedure to be followed to count function points. The COnstructive COst MOdel(COCOMO) is an algorithmic model utilized to anticipate software cost. It was produced by Barry Boehm in 1981, and was known as COCOMO’81. COCOMO depends on regression model.

R. T. Hughes has proposed a model based on expert judgment by a group of experts to utilize their experiences for estimation of a proposed software. The Delphi technique can be used to provide communication and cooperation among experts. One of the major drawbacks of the expert judgment model is the lack of analytical argumentation, because of the frequent use of phrases, which is identified . 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. Another approach is to utilize analogy based estimation strategy proposed by Shepperd et al. They have evaluated analogy approach with six distinct datasets drawn from a range of diﬀerent environments and their approach is being claimed to outperform other methods. The main disadvantage of analogy method is that it requires considerable amount of computation. Walkerden and Jeﬀery have compared few techniques for analogy-based software eﬀort estimation with each other furthermore with a linear regression model. The outcomes demonstrated that human brains work superior than tools at selecting analogies for the considered dataset. Estimates based on their selections, with a linear size adjustment in accordance with the analogue’s eﬀort esteem, demonstrated more precise results than estimates based on analogues selected by tools, furthermore more exact than evaluations based on the simple regression model. Idri et al. have proposed new and modified Analogy-based Software development Eﬀort Estimation (ASEE) techniques and the detailed analysis of result showed that ASEE methods outperform the eight techniques with which they were compared, and tend to yield acceptable results especially when combining ASEE techniques combines with Fuzzy Logic (FL) or Genetic Algorithms (GA). Idri et al. have also proposed a novel analogy-based technique, called 2FA-kprototypes, to foresee eﬀort when software projects are depicted by a blend of numerical and categorical attributes and coordinated fuzzy k-prototypes calculation into the procedure of estimation by analogy. The estimation precision of 2FA-kprototypes was assessed and contrasted with two techniques i.e., classical analogy-based technique and 2FA-kmodes utilizing four datasets. The outcomes acquired demonstrated that both 2FA-kprototypes and 2FA-kmodes perform superior than classical analogy-based technique.

Molokken and Jorgensen abridged estimation knowledge by conducting a survey on software eﬀort estimation. They found that most projects (60-80%) experience eﬀort and/or schedule overruns. The estimation techniques in most regular utilization of expert judgment is that there is no confirmation that formal estimation models lead to more correct assessments. The review likewise proposed that there is a lack of surveys including extensive analyses of the reasons for eﬀort and schedule overruns. Magne Jorgensen displayed seven rules for producing realistic software development eﬀort estimates, which are derived from industrial experience and observational studies. By dissecting the rules, they found that assessing eﬀort on the premise of expert judgment is the most widely recognized approach today, and the choice to utilize such procedures rather than formal estimation models can prompt more practical appraisals of software development eﬀorts. Kocaguneli et al. investigated the use of transfer learners for software eﬀort estimation, when project needs adequate local data in order to make accurate prediction. They have utilized dataset based on 154 number of projects collected from two diﬀerent sources to examine transfer learning between various time intervals and another dataset based on 195 number of projects collected from 51 diﬀerent sources to give a proof on the estimation of transfer learning for customary cross-company learning issues. From the examination of the outcome, it is found that transfer learning is a promising research direction that exchanges applicable cross data between time intervals and areas.

Whigham et al. have proposed an Automatically Transformed Linear Model (ATLM) as a reasonable baseline model for examination against software eﬀort estimation strategies. ATLM is a basic model, yet performs well over a range of various project types. Additionally, ATLM might be utilized with mixed numerical and categorical data and requires no parameter tuning. It is also deterministic in nature which means that results obtained are amenable to replication. They have suggested that ATLM should be used as a baseline of eﬀort prediction quality for all future model comparisons in SEE. Gonzalez et al. have performed a systematic mapping study over 107 number of papers that use International Software Benchmarking Standards Group (ISBSG) data for eﬀort estimation. They described the usage of ISBSG variables for filtering, as dependent variables, and as independent variables and identified 20 variables (out of 71) mostly used as independent variables for eﬀort estimation. By analyzing their study, they proposed guidelines for researchers to make informed decisions about which diﬀerent ISBSG variables to be selected for their eﬀort estimation models.

### Survey on Class Point Approach

### During the calculation procedure of adjusted class point as identified by Costagliola et al, two measures, Class Point 1 (CP1) and Class Point 2 (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). They have 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 have 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. have 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. have presented some new meanings of class point to interpret system’s architectural complexity in an improved way. They have utilized various additional parameters along with NEM, NSR and NOA to compute the total number of adjusted class point value.

Kanmani et al. have 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. Kapoor and Pandey have applied fuzzy logic technique along with class point approach for size estimation of object oriented products. Fuzzy logic allows a gradation of values instead of discrete sets, which in turn allows it to be more tolerant to uncertainty, imprecision, partial truth and approximation and thus achieve tractability, robustness and low cost solution. They have proved that fuzzy class point approach yields better results than traditional methods.

### Survey on Use Case Point Approach

### Issha et al. have investigated the evolution of three diﬀerent use case model-based software eﬀort estimation techniques. The correctness of the proposed techniques is verified using a wide range of software projects. Nassif et al. have presented two models i.e., log-linear regression (LLR) model and Multi-Layer Perceptron (MLP) model, based on use case point to compute the software development eﬀort focused around use case diagrams. By analyzing the outcome, they have proved that the MLP model performed better than other models for smaller projects; however, the LLR model outperforms other models for large size projects. Nassif et al. have presented a novel regression technique for eﬀort estimation of a given software focused around the UCP. They proposed a software eﬀort estimation equation that considers the non-linear relationship between software eﬀort and size, and in addition on the impacts of projects complexity and productivity. Results show that the accuracy of estimating the software development eﬀort gets enhanced by 16.5% than the result obtained using Karner’s model. Urbanek et al. have analyzed the statistical value of Use Case Points method parameters to find any parameters in Use Case Points method, which can be omitted from the calculation and the results may turn out to be better, while analytical programming for eﬀort estimation is being considered. From the result, it was observed that the accuracy of Use Case Points method is improved if and only if UUCW parameter is present in the calculation.

Nassif et al. have extended this process by applying Mamdani fuzzy inference system with regression model to enhance the estimation accuracy and found 10% improvement over Karners model and 6% over Schneiders model. Nassif et al. also applied Sugeno fuzzy inference system with regression model to enhance the estimation accuracy and found 11% improvement in the Mean Magnitude of Relative Error (MMRE) result over Karners model and 7% over Schneiders model. Nassif et al. have proposed an Artificial Neural Network (ANN) model to anticipate software eﬀort from use case diagrams based on the UCP model with the assistance of a dataset based on 240 number of projects and obtained an improved result than other regression models. A. B. Nassif has 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 over Karners model.

Nassif et al. have used a tree boost (Stochastic Gradient Boosting) model to estimate the eﬀort required to develop a software product focused around UCP method using a eighty four project dataset and achieved improved results. Saroha and Sahu have reviewed various techniques used for software eﬀort estimation and also provides comprehensive analysis of various tools and frameworks developed for eﬀorts estimation based on Use Case Point (UCP) model. They have observed that the analyzed tools provide opportunity to consider some other factors, which may aﬀect project delivery and help in providing a better estimate of project eﬀort than the existing ones. Silhavy et al. have presented a new size estimation method known as Algorithmic Optimisation method for estimate size of software engineering projects. This method is based upon use case point and multiple least square regression and is derived into three phases. They have observed that the proposed method performs approximately 43% better than use case point approach based on their magnitude of relative error score.

Periyasamy and Ghode have extended the original UCP model with additional information obtained from use case narratives. They classified actors into seven distinct groups. Additionally, the authors proposed new weights for use cases. The weight of an use case is resolved in light of the quantity of relationship amongst actors and the use cases. Anda et al. have provided guidance for other organizations who want to improve their estimation process applying use cases considering three industrial case studies. Results demonstrate that the direction gave by the use case point strategy can bolster expert knowledge in the estimation procedure and the configuration of the use case models strongly aﬀects the assessments. Sergey Diev has presented a number of real world situations taking into account the experience accumulated during deployment of the UCP in a product development department of a noteworthy financial establishment. The author exhibited that in order to get sensibly precise estimates, it is wanted to reflect in used case models a few aspects of the existing application and of the present project. Likewise, the author recommended a few elucidations of the idea of use case transaction and frameworks some approaches to bolster use case models consistency inside and crosswise over projects.Ajitha et al. have built up a neural system model to evaluate the size of software utilizing Use Case Point approach. The outcomes are approved and a contextual analysis of Multi-Agent System and showed improvement over the existing ones. Iraji and Motameni have presented an adaptive fuzzy neural network model to estimate the eﬀort of object oriented software using Use Case size Point approach. Results indicate that the proposed approach possesses less error and worked more accurately than methods evolved earlier. Nassif et al. have proposed an approach to calibrate the complexity weights of the use cases in the Use Case Points (UCP) model. They applied a neural network with fuzzy logic to tune the complexity weights. Saroha and Sahu have proposed an enhanced Use Case Point model (Algorithmic model) to overcome the problems arise from using the existing Use Case Point (UCP) model by considering five software projects case studies and diﬀerent evaluation criteria like MMRE, MMER, MSE, RMSE and PRED(x). It was evident that the result obtained from the proposed model outperforms the results obtained from existing UCP model.

### 2.4 Survey on Eﬀort Estimation of Web Applications

### Mendes et al. have worked extensively on the aspects of size-measurements and cost drivers for early stage web eﬀort estimation by taking the help of dataset based on 133 number of web projects. Results showed that the two most basic size measurements utilized for web eﬀort estimation were “total number of web pages” (70%) and “which functionality to be given by the application”(66%). Emilia Mendes has employed four diﬀerent approaches for estimation of eﬀort required to develop web-based applications: Forward Stepwise Regression (FSR), Bayesian Networks (BNs), Case Based Reasoning (CBR) and Classification and Regression Trees (CART) to get the estimated eﬀort and compared them. Mendes and Mosley have also compared several BN models using a cross-company dataset for eﬀort estimation in web developments. The developed models’ various performance parameters were also compared to mean and median-based eﬀort models, MSR and CBR. Corazza et al. have investigated the use of Tabu Search meta-heuristic methodology in blend with SVR to choose a suitable subset of parameters to be utilized for web eﬀort estimation using the same database and obtained promising results.

Ferrucci et al. have inquired the viability of Tabu Search in assessing the eﬀort required to develop web-based applications with the help of Tukutuku cross-company database and obtained encouraging results. Elyassami and Idri have investigated the eﬀectiveness of applying Fuzzy ID3 decision tree technique for software eﬀort estimation purpose. This technique is outlined by incorporating the standard principles of fuzzy set-theoretic concepts into the ID3 decision tree. Corazza et al. have investigated the viability of SVR for web eﬀort estimation utilizing a cross-company dataset and thought about diverse SVR designs taking a gander at the particular case that exhibits the best execution. The dataset utilized for validation was the Tukutuku database and results demonstrated that the SVR RBF outperforms others. Martino et al. have enquired the potency of the Web Objects measure as an indicator of We-based software development eﬀort. The eﬀectiveness of the Web Objects measure as indicator of Web application development eﬀort was confirmed, when assembled with Ordinary Least-Squares Regression (OLSR) and WebCOBRA, and this is true even when using CBR. It was observed that the Web Objects method yields better results than the FPA method when assembled with OLSR and Web-COBRA.

Ferrucci et al. have studied the suitability of web eﬀort estimation models developed with the help of cross-company dataset and compared it with the model based on single-company dataset. They have performed the validation of the models considering dataset based on 195 number of web projects obtained from the Tukutuku database as input. Results proved that although the prediction accuracy value of the model obtained using cross-company dataset was not more impressive than that of single-company models; but by applying the filtering mechanism, the prediction accuracy value can be improved significantly. Corona et al. have presented a new methodology for developing a web eﬀort estimation model with a content management framework (CMF) and performed the experimental validation using dataset of nine numbers of projects as provided by three diﬀerent Italian software companies.

Kocaguneli et al. have presented the eﬀectiveness of applying ensembles of eﬀort estimation techniques. They generated ninety number of solo methods, which were applied to twenty number of datasets and the results were evaluated using seven numbers of error measures. It is observed that authors have combined few solo methods to generated twelve number of multi methods. From the analysis of the result, they observed that no single eﬀort estimation method can be identified to be the best, but there exists a suitable combination of such eﬀort estimation methods, which may yield better results. Azhar et al. have presented the use of ensembles of eﬀort estimation techniques for web project data using two approaches i.e., replication of methodology and using Scott-Knott algorithm. The replication identified 16 number of techniques out of 90 number of solo estimation techniques on web project data from the Tukutuku dataset to build 15 ensembles; whereas the Scott-Knott algorithm identified 19 superior solo techniques that were used to build two ensembles. Results showed that ensembles of techniques outperformed solo estimation technique. The study carried out by Azhar et al. is a production study of Kocaguneli’s work. Matos et al. have performed information examination utilizing Grounded Theory-based techniques to distinguish and join components influencing the eﬀort estimation process of web applications and recognized four categories of factors. The factors available in each of these groupings aﬀects the process of estimating eﬀort for web applications. Matos et al. have also extended the work to build the comprehension of web eﬀort estimation by utilizing the same set of factors already identified in the previous article alongside the knowledge from experts to handle eﬀort estimation process. They have recognized a sum of 90 number of variables which make an impact on eﬀort estimation in web applications, out of which only 30 number of components were distinguished during extensive research study carried out with experts Nassif et al. have provided a comparative analysis of results obtained by applying four diﬀerent neural network models such as Multi-layer Perceptron (MLP), General Regression Neural Network (GRNN), Radial Basis Function Neural Network (RBFNN) and Cascade Correlation Neural Network (CCNN) for software eﬀort estimation purpose. They observed that CCNN model outperformed other models based on 60 % of the dataset, where as RBFNN outperforms other models based on 40% of the datasets. They also proved that CCNN model is not statistically diﬀerent from other models despite of its higher performance. Denis and Boris have analyzed the possibility of using a combination of functional size and conceptual models for the purpose of web application development eﬀort estimation. They have employed 19 web applications with their conceptual models to build an eﬀort model using simple linear regression analysis and obtained promising results for web projects used in the model construction and validation process. Barabino et al. presented a new methodology, called *Web Framework Points*, in order to assess the eﬀort required to develop web-based applications with Content Management Framework (CMF). They performed the validation of the results obtained from the research work by using dataset based on 29 number of projects, out of which 83% showed less than 25% of estimation error value.

Bhardwaj and Rana portrayed the connections among size of a software, no. of software defects, productivity and eﬀorts for web applications established utilizing the multiple linear regression technique on the data collected from ISBSG. Results suggest that in web-based projects the number of defects identified is directly proportional to the productivity. Therefore, less testing and rework eﬀort will be required if project is planned with lower productivity. Minku et al. examined the utilization of Dycom approach to evaluate to what degree Web eﬀort estimation acquired utilizing cross-comapny (CC) datasets are viable in connection to the predictions got utilizing within-company (WC) data when unequivocally mapping the CC models to the WC context. A 125 number of web-based projects data 25 from eight distinctive organizations part of the Tukutuku database were utilized to build prediction models. They have benchmarked these models against baseline models (mean and median eﬀort) and a WC base learner that does not advantage of the mapping. By dissecting the outcomes, it was evident that Dycom mostly accomplish comparable or preferred execution over a WC model while utilizing just 50% of the WC training data. Martino et al. have empirically investigated the eﬀectiveness of COSMIC over FPA for Web eﬀort estimation. Two experimental studies have been performed by with the help of an industrial data set. Aftereﬀect of the principal study uncovered that, COSMIC was essentially more exact than FPs in assessing the development eﬀort for considered dataset. The second study uncovered that the viability of the investigated two-stage process fundamentally relies on the utilized conversion equation.

### 2.5 Survey on Story Point Approach for Agile Software Eﬀort Estimation

Keaveney and Conboy have investigated the applicability of conventional estimation techniques towards agile development approaches by underscoring on the case studies of agile method utilized within diverse organizations. Coelho and Basu have described the steps followed in story point-based method for eﬀort estimation of agile software and highlighted the areas which need to be looked into for further research. Andreas Schmietendorf et al. have provided an investigation about estimation possibilities, especially for the extreme programming paradigm.

Ziauddin et al. have developed an eﬀort estimation model for agile software projects, where model was fine-tuned with the help of the empirical data acquired from twenty one software projects. Usman et al. have provided a detailed overview of the state of the art in the area of eﬀort estimation in agile software development. They considered 25 primary studies for review purpose and identified several research gap relating to the agile methods, size metrics and cost drivers. Hearty et al. have proposed a Bayesian network model of an XP surrounding and indicated how it could gain from project data keeping in mind the end goal to predict the eﬀort and risk appraisals without obliging any extra metrics.

Popli and Chauhan have proposed a model for eﬀort and cost estimation in agile software development by using regression analysis. Hussain et al. have made an attempt to propose an approach which helps in removing problems like formalized user requirements and thus apply function points for agile software eﬀort estimation. A. E. D. Hamouda have introduced a process and methodology that guarantees relativity in software sizing while using agile story points. This proposed process and methodology was applied in a Capability Maturity Model Integration (CMMI) level three company on diﬀerent projects. Ungan et al. have compared SCRUM’s native eﬀort estimation method Story Points and poker planning, with eﬀort estimation models based on COSMIC Function Points (CFP) for a selection of projects. by using regression models and ANN methodology and proved that COSMIC measurement is a better method for eﬀort estimation than SCRUM’s story points.

Viljan Mahnic have described a case study with the aim of studying the behavior of development teams utilizing scrum for the first time, i.e., a situation typical for software companies attempting to bring scrum into their development process. It was found that the initial plans and eﬀort estimates were over-optimistic, but the abilities of estimating and planning improved from sprint to sprint. Mahnic and Zabkar have extended their approach by describing a set of measures that give IT administration with continuous understanding in the scrum-based software development process. The proposed measures were applied within the scope of the project of rebuilding a web site, which served as a contextual investigation for assessment of their ease of use. The contextual investigation demonstrated that each proposed measure depicts a valuable process aspect and collection of data does not require additional administrative work that would harm the agility of scrum. Garg and Gupta have applied Principal Component Analysis (PCA) to reduce the dimensions of the attributes required and identify the key attributes which have maximum correlation to the development cost; and then use constraint solving approach to satisfy the criteria imposed by agile manifesto. The proposed methodology is found to bet suitable for agile projects as it uses constraint programming to explicitly check for satisfaction of agile manifestos. From the analysis of results, it is found that the proposed model exhibits a low MMRE value than the existing models.

Lenarduzzi et al. have introduced functional size metrics to improve estimation accuracy and to measure the accuracy of expert-based estimation. Further they extended this approach to plain Scrum processes, where the original study was replicated twice, applying an exact replication to two plain Scrum development processes. The results of this replicated study show that the accuracy of the eﬀort estimated by the developers is very accurate and higher than that obtained through functional size measures. Raslan et al. have proposed a framework based on the fuzzy logic which receives fuzzy input parameters of Story Points (SP), Implementation Level Factor (ILF), Friction factors (FR), and Dynamic Forces (DF) to be processed in many successive steps to produce in final the eﬀort estimation. They analyzed the utilization of fuzzy logic in improving the eﬀort estimation accuracy using the user stories by characterizing inputs parameters using trapezoidal membership functions. Britto et al. have performed an empirical investigation on the state of the practice on eﬀort estimation in AGSD. To do so, a survey was carried out using as instrument an on-line questionnaire and a sample comprising software practitioners experienced in eﬀort estimation within the Agile Global Software Development (AGSD) context. Results show that the eﬀort estimation techniques used within the AGSD and collocated contexts remained unchanged, with planning poker being the one employed the most.

### 2.6 Survey on Software Eﬀort Estimation using Machine Learning

### Techniques

Alaa Sheta has used Takagi-Sugeno-Kang (TSK) fuzzy model to develop fuzzy models for two diﬀerent type of nonlinear processes. The first one is based on NASA software projects eﬀort estimation process and the second one is on the stock market prediction process for S & P 500. Kocaguneli et al. have investigates diﬀerent software eﬀort estimation techniques and found that no single technique reliably beats all others. Subsequently, it is more astute to generate estimates from gatherings of various estimation techniques. Elyassami and Idri have 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. Pahariya et al. have presented a novel Genetic Algorithm (GA)-based feature selection algorithm for estimating the eﬀort required to develop a software and compared the result with the output obtained using other ML techniques. Results proved that the proposed technique outperformed all the other existing techniquesAdriano L.I. Oliveira have provided a comparative study on support vector regression (SVR), radial basis function neural networks (RBFNs) and linear regression for estimation of software project eﬀort. The experiment is carried out using NASA project datasets and the result shows that SVR performs better than RBFN and linear regression. Braga et al. have proposed and investigated the use of a genetic algorithm approach for selecting an optimal feature subset and optimizing SVR parameters simultaneously aiming to improve the precision of the software eﬀort estimates. Kocaguneli et al. have investigated non-uniform weighting through kernel density estimation and found that nonuniform weighting through kernel methods cannot outperform uniform weighting Analogy Based Estimation (ABE). Bakele et al. have proposed ML technique-based eﬀort estimation models and assess the models by taking the help of publicly available resources and data accumulated from software industries. By analyzing the results, it is observed that the utilization of any one model for software eﬀort estimation purpose may not always provide the best possible solution.

Wen et al. have intended to eﬃciently dissect machine learning models from four viewpoints such as type of machine learning technique, accuracy of estimates, comparison of model, and the context of estimation by taking the help of 84 primary studies. They have observed that the estimation of accuracy obtained using machine learning models is near the satisfactory level and is superior to anything that of non-machine learning models. Azzeh et al.have integrated analogy-based estimation with Fuzzy numbers in order to improve the performance of software project eﬀort estimation during the early stages of a software development life cycle, using all available early data. Results inferred that the proposed similarity measure and adaptation techniques method were able to significantly improve the performance of analogy-based estimation during the early stages of software development and outperform the results of Case-based Reasoning (CBR) and Stepwise Regression.

Azzeh et al. have investigated the potential of ensemble learning for variants of adjustment methods used in analogy-based eﬀort estimation. They performed a large scale comparison study where many ensembles constructed from n out of 40 possible valid variants of adjustment methods are applied to eight datasets. The results have been subjected to statistical significance testing, and show reasonable significant improvements on the predictive performance where ensemble methods are applied. Mendes et al. have described an industrial case study where an expert-based requirements eﬀort estimation model was built and validated for the Brazilian Navy. A knowledge engineering of Bayesian networks process was employed to build the requirements eﬀort estimation model. The expert-based requirements eﬀort estimation model was built with the participation of seven software requirements analysts and project managers, leading to 28 number of prediction factors and more than 30 number of relationships. The model was validated based on real data from 11 number of large requirements specification. The model was incorporated into the Brazilian navys quality assurance process to be used by their software requirements analysts and managers.

### Summary of Observations

### For eﬀort estimation process, it can be observed that function point and COCOMO has been used by a number of researchers as input in design of prediction models. It is observed that the public datasets are also most commonly used in object oriented software eﬀort estimation. Traditional software estimation techniques like Constructive Cost Estimation Model (COCOMO) and Function Point Analysis (FPA) have been proved to be unsatisfactory for measuring the cost and eﬀort of all types of software development. This is because the line of code (LOC) and function point (FP) were both used for procedural programming concept. The procedural oriented design splits the data and procedure, whereas the object oriented design combines both of them. But, as Unified Modeling Language (UML) diagrams become a popular approach to represent object-oriented softwares, the use of class point approach derived from Class Diagram and Use Case point approach based on the requirement analysis phases as well as from use case diagram are the solutions, which will have wider acceptance for object-oriented software eﬀort estimation purpose. With the rising use of dependency on Web, there is a necessity of quick and eﬃcient development of web-based software. For developing web-based software eﬃciently i.e. without any cost or resource (human or otherwise) overrun, the estimates that are done before the beginning of development need to be correct. It is noticed that many authors have used Tukutuku dataset for web eﬀort estimation process, which is not publicly available. Very few authors have used the ISBSG dataset for web eﬀort estimation. Hence, by more exploring the application of ISBSG dataset for web eﬀort estimation process and improving the prediction accuracy by using statistical and machine learning techniques, it will provide more flexibility as well as better prediction accuracy for the industries to estimate the eﬀort of web applications development.

### Agile software eﬀort estimation is also one of the promising area of research. Many researchers have proposed various methodologies for agile software development process. But there is lack of availability of good amount of research work for providing a systematic procedure in order to estimate the eﬀort of softwares developed using agile methodology. Story point approach is one of the popular ways to estimate the eﬀort of softwares developed using scrum methodology. But there is a very scarcity of dataset based on story point approach due to unavailability of project velocity information. Hence, by applying the diﬀerent statistical and machine learning techniques on the considered story point dataset, the accuracy of agile software eﬀort estimation process will be improved.

**2.8 EXISTING SYSTEM**

### Software effort time estimation using linear regression is a technique for predicting the amount of time required to develop a software project. It involves using a statistical method called linear regression to fit a linear model to historical data on software projects. The model can then be used to estimate the effort for new projects.

### Steps in using linear regression for software effort time estimation:

### Collect historical data: Gather data on past software projects, including the effort (in hours or person-months) required to develop each project and a set of independent variables that are thought to be related to effort. These variables could include the size of the project, the complexity of the project, and the experience of the development team.

### Data preparation: Clean and prepare the data for analysis. This may involve handling missing values, transforming variables, and removing outliers.

### Split the data: Divide the data into two sets: a training set and a test set. The training set will be used to fit the linear model, and the test set will be used to evaluate the model's performance.

### Fit the linear model: Use a statistical software package to fit a linear regression model to the training data. The model will provide coefficients for each of the independent variables, which represent the estimated impact of each variable on effort.

### Evaluate the model: Use the test data to assess the performance of the linear regression model. This can be done by calculating metrics such as mean absolute error (MAE) and root mean squared error (RMSE).

### Advantages of using linear regression for software effort time Estimation:

### Simplicity: Linear regression is a relatively simple and easy-to-understand technique.

### Interpretability: The coefficients of the linear regression model can be interpreted to understand how each independent variable affects effort.

### Efficiency: The linear regression model can be trained and evaluated efficiently, even with large datasets.

### Limitations of using linear regression for software effort time estimation:

### Linearity assumption: Linear regression assumes that the relationship between effort and the independent variables is linear. This assumption may not hold for all software projects.

### Overfitting: If the model is too complex, it may overfit the training data and not generalize well to new projects.

### Multicollinearity: If there is multicollinearity among the independent variables, it can make it difficult to interpret the coefficients of the model.

### Overall, linear regression can be a useful technique for software effort time estimation, but it is important to be aware of its limitations. Other techniques, such as nonlinear regression or machine learning algorithms, may be more appropriate for certain datasets.

**CHAPTER 3**

**CLASS POINT APPROACH FOR SOFTWARE EFFORT ESTIMATION USING MACHINE LEARNING TECHNIQUES**

**3.1 Introduction**

Object-oriented (OO) technology is the accepted methodology in the present day scenario for software development in major industries as it helps in building software development process in a more organized fashion. 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 Approach (FPA) and COCOMO, cannot be claimed as the most eﬃcient techniques to estimate the cost and eﬀort required to develop the software These techniques are not capable of measuring eﬃciently the cost and eﬀort, because they are tailored for procedural-oriented software development paradigm. The procedural oriented paradigm and object-oriented paradigm diﬀer because the former splits the data and procedure; while the later combines them.

It is important to realize that the problem of learning/estimation of dependencies from samples is only one part of the general experimental procedure used by researchers and practitioners who apply statistical methods to draw conclusions from the data. Hence to obtain proper results in estimating software size, it is essential to consider the data obtained from previous projects. As far as eﬀort estimation is concerned, a number of unsolved problems and errors still exist. Estimation of a software project is always important aspect for determining the feasibility of the project. In the present scenario, most of the software project planning activities depend upon estimated figures of eﬀort. Since *class* is the fundamental logical unit of an OO system, the utilization of the class point methodology to compute the project eﬀort serves as a basic guideline. During the calculation procedure of the final adjusted class point, two measures, Class Point 1 (CP1) and Class Point 2 (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 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. Hence in this study, the optimized CPA is utilized to ascertain the eﬀort needed to create the software adopting these six non-technical factors. Likewise with a specific end goal to accomplish an improved value of prediction accuracy, Stochastic Gradient Boosting (SGB) and four Support Vector Regression (SVR) Kernels-based eﬀort estimation models are applied over the obtained class point value. The results obtained from the these models are then compared with the results obtained from other machine learning techniques available in literature in order to access their performance.

### 3.2 Methodologies Used

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

#### Class Point Approach (CPA)

#### The CPA was presented by Costagliola et al. in 1998. It is focused around the FPA methodology to speak to the interior qualities of a software. The essential thought of the CPA system is calculation of number of classes in a project. It is derived 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 4.1, displays the steps to compute the project development eﬀort using class point approach.

The system to acquire the amount of class points is isolated into three principal

* + Estimation of information processing size
    - Identification and classification of classes
    - Evaluation of complexity level for each classified class
    - Calculation of the Total Unadjusted Class Points (TUCP) value
  + Estimation of the Technical Complexity Factor (TCF) value
  + Calculation of the final value of Adjusted Class Point (ACP)

**Class Diagram**

**Identification and**

**Classification of Classes**

**Evaluation of Final Adjusted Class Point**

Figure 3.1 Steps to Calculate Final Adjusted Class points.

**Calculation of TUCP and TCF**

**Assignment of Complexity Level to Each Class**

**Identification and Classification of Classes**

During the first step, the design specifications are analyzed in order to identify and classify the classes into four types of system components, namely the Problem Domain Type (PDT), the Human Interaction Type (HIT), the Data Management Type (DMT), and the Task Management Type (TMT). The PDT component contains classes representing real-world entities in the application domain of the system. The classes of HIT type are designed to satisfy the need for information visualization and human-computer interaction. The DMT component encompasses the classes that oﬀer functionality for data storage and retrieval. Finally, TMT classes are designed for task management purposes, thus they are responsible for the definition and control of tasks.

##### Evaluation of complexity level for each classified class

##### During the second step, each identified class is assigned a complexity level, which is determined on the basis of methods associated with the class and of the interaction of the class with the rest of the system. In some cases, the complexity level of each class is determined on the basis of the NEM, and NSR. In some other cases, besides the above measures, the NOA measure is taken into account in order to evaluate the complexity level of each class. For the calculation of CP1, the complexity level of the class is determined based on the value of NEM and NSR according to Table 4.1. For example, if a class is having NEM value 7 and NSR value 3, then the complexity level can be assigned to the class as ‘Average’.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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 |

**Table 3.1** **Complexity Level Evaluation for CP1**

For the calculation of CP2, the complexity level of the class is determined based on the value of NEM, NOA and NSR according to Table 3.2. From Table 3.2, it is observed that ranges of NEM and NOA vary with respect to fixed NSR range.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***0 - 2 NSR*** | **0 - 5 NOA** | **6 - 9 NOA** | **10 - 14 NOA** | ≥ 15 **NOA** |
| **0 - 4 NEM** | Low | Low | Average | High |
| **5 - 8 NEM** | Low | Average | High | High |
| **9 - 12 NEM** | Average | High | High | Very High |
| ≥ 13 **NEM** | High | High | Very High | Very High |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***3 - 4 NSR*** | **0 - 4 NOA** | **5 - 8 NOA** | **9 - 13 NOA** | ≥ 14 **NOA** |
| **0 - 3 NEM** | Low | Low | Average | High |
| **4 - 7 NEM** | Low | Average | High | High |
| **8 - 11 NEM** | Average | High | High | Very High |
| ≥ 12 **NEM** | High | High | Very High | Very High |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0 - 3 NOA** | **4 - 7 NOA** | **8 - 12 NOA** | ≥ 13 **NOA** |
| **0 - 2 NEM** | Low | Low | Average | High |
| **3 - 6 NEM** | Low | Average | High | High |
| **7 - 10 NEM** | Average | High | High | Very High |
| ≥ 11 **NEM** | High | High | Very High | Very High |

≥ 5 ***NSR***

**Table 3.2**: **Complexity Level Evaluation for CP2**

##### Calculation of the Total Unadjusted Class Points (TUCP) value

##### Once a complexity level of each class has been assigned, such information and its type are used to assign a weight to the class given in Table 3.3. Then, the Total Unadjusted

##### Class Point value (TUCP) is computed as a weighted sum as shown below:

4 3

*TUCP* = ∑ ∑ *wij* × *xij*

*j*=1

*i*=1

(3.1)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 |

where *xij* is the number of classes of component type i (problem domain, human interaction, etc.) with complexity level j (low, average, or high), and *wij* is the weighting value for type i and complexity level j.

Table 3.3: Evaluation of TUCP Value for Each Class Type

##### Estimation of the Technical Complexity Factor (TCF) value

##### The Technical Complexity Factor (TCF) is determined by adjusting the TUCP with a value obtained by 24 diﬀerent target software system characteristics, each on a scale of 0 to 5. The sum of the influence degrees related to such general system characteristics forms the Total Degree of Influence (TDI) as shown in Table 3.4, which is used to determine the TCF according to the following formula:

*TCF* = 0*.*55 + (0*.*01 ∗ *TDI*)

(3.2)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *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 | Multiuser Interactivity | .... |
| C6 | Online data entry | .... | C18 | Multiple Interfaces | .... |
| C7 | End-user eﬃciency | .... | C19 | Management Eﬃciency | .... |
| C8 | Online update | .... | C20 | Developers’ Professional Competence | .... |
| C9 | Complex processing | .... | C21 | Security | .... |
| C10 | Reusability | .... | C22 | Reliability | .... |
| C11 | Installation ease | .... | C23 | Maintainability | .... |
| C12 | Operational ease | .... | C24 | Portability | .... |
| TDI | Total Degree of Influence (TDI) | | | | .... |

Out of all the twenty-four characteristics, some of them are very important for object-oriented systems such as user adaptivity, rapid prototyping, multiuser interactivity, multiple interface. Two characteristics i.e., management eﬃciency and developers’ professional competence help for calculation of CP2. These two are not considered for CP1 calculation. The complex processing characteristic describes the complexity level of the tasks. Similarly, developers’ professional competence characteristic describes the technology and skills, the developer of project possess. If the developers’ have good professional competence, then it will be easier for them to develop the project eﬀectively. The facilitation of change characteristic denotes the ability to adopt changes quickly; whereas maintainability characteristic denotes a number of aspects such as isolation, correction and prevention of defects, maximization of products life cycle, maximization of eﬃciency, reliability and security of software project etc.

**Table 3.4 Degree of Influences of 24 General System Characteristics**

##### Calculation of the final value of Adjusted Class Point (ACP)

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

*ACP* = *TUCP* ∗ *TCF*

(3.3)

In this study, the above phases are followed to calculate final optimized class points. Herein, the total number of class point value is then used as an input parameter to the ML techniques-based eﬀort estimations models to calculate the estimated eﬀort.

### CHAPTER 4

### PROPOSED SYSTEM

### The proposed approach is tested by using dataset that contains data derived from forty projects, which were developed using the Java language. The dataset is used to evaluate software development eﬀort and to validate the improvement. The results obtained in the validation process provided initial experimental evidence of the eﬀectiveness of CPA. The dataset is used to develop the SGB and diﬀerent SVR kernel-based software eﬀort estimation model. The block diagram, shown in figure 4.1, displays the proposed steps used to determine the predicted eﬀort using the SGB and four SVR kernel techniques. To calculate the eﬀort of a given software project, the following steps are used.



**Performance Evaluation**

**Performing Model Selection**

**Division of Dataset**

**Normalization of Dataset**

**Calculation of Class Points**

**Figure 4.1 Proposed Steps Used for the Eﬀort Estimation based on CPA using SGB and SVR Kernel Techniques**

##### Steps in Eﬀort Estimation

1. **Calculation of Class Points**: In this step, the CP1 and CP2 values are calculated from the class diagram. Generated CP1 and CP2 values are used as an input argument.
2. **Normalization of Dataset**: Input parameter values are individually normalized to the range. Let X be the dataset and *x* be an element of the dataset. Then, the normalized value of *x* can be calculated as :

Normalization(x)= x-min(x)

Max(x)-min(x)

min(*X*) represents the minimum value of the dataset *X* and max(*X*) represents the maximum value of the dataset *X*. If max(*X*) is equal to min(*X*), then Normalized(*x* ) is set to 0.5.

1. **Division of Dataset**: The dataset is divided into three parts, such as learning set, validation set and test set.
2. **Performing Model Selection**: To select eﬀort estimation model based on SGB technique, first the values of various parameters such as number of trees, Huber’s quantile cutoﬀ, shrinkage factor, stochastic factor and influence trimming factor are found out and then a five-fold cross validation is implemented for model selection. The advantage of using cross validation is to avoid the possible biasness introduced by relying on any particular division of dataset for training and testing. It provides the advantage of using all data for training and testing purpose. The model that provides the lowest RMSE,

MAE, MMRE, MMER values and the highest prediction accuracy (PRED (x)) values is selected as the best model for each fold.

Similarly, in case of SVR kernel-based eﬀort estimation model, the model which provides the lesser value than the other generated models based on the minimum validation error criteria has been selected to perform other operations. The tunable parameters are selected to find the most suitable parameters of C and *γ* using a five-fold cross validation procedure. Based on the minimum validation error, the best model has been selected and the corresponding values of *γ* as well as *ϵ* are found out. The final model selected based on best parameter of C, *ϵ* and *γ* have been trained using all training samples. The output of this step is the trained SVM model providing predicted response values for test inputs.

1. **Performance Evaluation**: The performance of the model is verified using final RMSE, MAE, MMRE, MMER and PRED(x) values obtained from test samples. The obtained values of various models are compared with the existing results as well as among themselves in order to access their accuracy.

The above steps are followed to implement the SGB and SVR Kernel-based eﬀort estimation models. Finally, a comparison of the results obtained using these models with the results obtained from the other models is presented with an objective to assess their performances.

### 4.1 Experimental Details

### In the proposed research study, the dataset collected from Costagliola et al. and Zhou and Liu, listed in Tables 4.1 and 4.2 respectively, are used. In these tables, 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.

The statistical profile of two diﬀerent categories of dataset collected for Class Point Approach is depicted in Table 4.3 From this table, it can be observed that both the 40 and 30 projects datasets are more normally distributed based on the values of the skewness and kurtosis.

Figures 3.3a and 3.3b depict the relationship between software size and software eﬀort (person-hours) based on CP1 & CP2, using the dataset of 40 projects. Similarly, Figures 3.3c and 3.3d depict the relationship between software size and software eﬀort (person-hours) based on CP1 & CP2 using 30 project dataset respectively.

From these figures, it is observed that the 30 project dataset contains more number of outliers than 40 project dataset. Figures 3.4a and 3.4b display the histogram of eﬀort value for 40 and 30 project dataset respectively. From these figures, it can been observed that 40 project dataset are more normally distributed based on the values of the skewness and kurtosis than 30 project dataset, which can also be verified based on values of skewness and kurtosis provided in Table 3.7,After calculating the final class point values, the dataset is then normalized.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **EFH** | **CP1** | **CP2** |
|  |  |  |  |
| 1 | 286 | 103.18 | 110.55 |
| 2 | 396 | 278.72 | 242.54 |
| 3 | 471 | 473.90 | 446.60 |
| 4 | 1016 | 851.44 | 760.96 |
| 5 | 1261 | 1263.12 | 1242.60 |
| 6 | 261 | 196.68 | 180.84 |
| 7 | 993 | 178.80 | 645.60 |
| 8 | 552 | 213.30 | 208.56 |
| 9 | 998 | 1095.00 | 905.00 |
| 10 | 180 | 116.62 | 95.06 |
| 11 | 482 | 267.80 | 251.55 |
| 12 | 1083 | 687.57 | 766.29 |
| 13 | 205 | 59.64 | 64.61 |
| 14 | 851 | 697.48 | 620.10 |
| 15 | 840 | 864.27 | 743.49 |
| 16 | 1414 | 1386.32 | 1345.40 |
| 17 | 279 | 132.54 | 74.26 |
| 18 | 621 | 550.55 | 481.66 |
| 19 | 601 | 539.35 | 474.95 |
| 20 | 680 | 489.06 | 438.90 |
| 21 | 366 | 287.97 | 262.74 |
| 22 | 947 | 663.60 | 627.60 |
| 23 | 485 | 397.10 | 358.60 |
| 24 | 812 | 678.28 | 590.42 |
| 25 | 685 | 386.31 | 428.18 |
| 26 | 638 | 268.45 | 280.84 |
| 27 | 1803 | 2090.70 | 1719.25 |
| 28 | 369 | 114.40 | 104.50 |
| 29 | 439 | 162.87 | 156.64 |
| 30 | 491 | 258.72 | 246.96 |
| 31 | 484 | 289.68 | 241.40 |
| 32 | 481 | 480.25 | 413.10 |
| 33 | 861 | 778.75 | 738.70 |
| 34 | 417 | 263.72 | 234.08 |
| 35 | 268 | 217.36 | 195.36 |
| 36 | 470 | 295.26 | 263.07 |
| 37 | 436 | 117.48 | 126.38 |
| 38 | 428 | 146.97 | 148.35 |
| 39 | 436 | 169.74 | 200.10 |
| 40 | 356 | 112.53 | 110.67 |

**Table 4.1 Forty Project Dataset**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **EFH** | **CP1** | **CP2** |
| 1 | 286 | 103.18 | 110.55 |
| 2 | 396 | 278.72 | 242.54 |
| 3 | 471 | 473.90 | 446.60 |
| 4 | 1016 | 851.44 | 760.96 |
| 5 | 1261 | 1263.12 | 1242.60 |
| 6 | 261 | 196.68 | 180.84 |
| 7 | 993 | 178.80 | 645.60 |
| 8 | 552 | 213.30 | 208.56 |
| 9 | 998 | 1095.00 | 905.00 |
| 10 | 180 | 116.62 | 95.06 |
| 11 | 482 | 267.80 | 251.55 |
| 12 | 1083 | 687.57 | 766.29 |
| 13 | 205 | 59.64 | 64.61 |
| 14 | 851 | 697.48 | 620.10 |
| 15 | 840 | 864.27 | 743.49 |
| 16 | 1414 | 1386.32 | 1345.40 |
| 17 | 279 | 132.54 | 74.26 |
| 18 | 621 | 550.55 | 481.66 |
| 19 | 601 | 539.35 | 474.95 |
| 20 | 680 | 489.06 | 438.90 |
| 21 | 366 | 287.97 | 262.74 |
| 22 | 947 | 663.60 | 627.60 |
| 23 | 485 | 397.10 | 358.60 |
| 24 | 812 | 678.28 | 590.42 |
| 25 | 685 | 386.31 | 428.18 |
| 26 | 638 | 268.45 | 280.84 |
| 27 | 1803 | 2090.70 | 1719.25 |
| 28 | 369 | 114.40 | 104.50 |
| 29 | 439 | 162.87 | 156.64 |
| 30 | 491 | 258.72 | 246.96 |

**Table 4.2 Thirty Project Dataset**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Type** | **Minimum** | **Maximum** | **Mean** | **Median** | **Std. Dev.** | **Skewness** | **Kurtosis** |
| **40 Project Dataset** | 180 | 1803 | 628.55 | 484.5 | 351.43 | 1.37 | 2.11 |
| **30 Project Dataset** | 55 | 2895 | 787.87 | 187 | 941.97 | 0.92 | -0.86 |

**Table 4.3 Statistical Profile of Two Datasets used for Class Point Approach**

**Effort**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

**0.5**

**0.4**

**0.3**

**0.2**

**0.1**

**0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1**

**Size (CP1)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

**0.7**

**0.6**

**0.5**

**0.4**

**0.3**

**0.2**

**0.1**

**0**

**0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1**

**Size(cp1)**

Figure 3.3: Software Size vs. Eﬀort Graph based on CP1 & CP2 using 40 and 30 Project Datasets

10 12

**Frequency**

**Frequency**

8 10

6 8

6

4

4

2 2

0

−500 0 500 1000 1500 2000

**Effort**

(a)

−3000 −2000 −1000 0 1000 2000 3000 4000

**Effort**

(b)

Figure 3.4: Histogram of Eﬀort Values for 40 and 30 Project Dataset

The normalized dataset is divided into diﬀerent subsets using a double sampling procedure. In the sampling procedure, the normalized dataset is first divided into a *training set* and a *test set*. The *training set* is used for learning purposes (model estimation), whereas the *test set* is used only for estimating the prediction accuracy of the final model. In the second step, the *training set* is divided into a *learning set* and a *validation set*. The *learning set* is used to estimate model parameters, and the *validation set* is used for selecting an optimal model (usually via cross validation).

Every fifth row of the Tables 3.5 and 3.6, 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 extracted for validation purposes, and the rest are used for learning purposes. Hence, after completion of the second step of the double sampling process, the complete thirty-two project dataset is divided into twenty-six rows for learning and six rows for validation. After partitioning the sample into a learning set and a validation set, the model selection is performed using a five fold cross validation process.

**4.1.1 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 Huber’s Quantile Cutoﬀ factor are transfered. 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 Huber’s 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 chosen to predict the eﬀort using the SGB technique.

No of Trees: 1000

Huber’s Quantile Cut oﬀ: 0.95

Shrinkage Factor: 0.1

Stochastic Factor: 0.5

Influence Trimming Factor: 0.01

The detailed descriptions of these parameters were already provided in Section 1.4.2. The values for the parameters are assigned by choosing appropriate combinations of values to generate the best result for the SGB-based eﬀort estimation model.

Figures 3.5a and 3.5b display the actual eﬀort and the predicted eﬀort obtained for CP1 and CP2 respectively using the SGB technique, taking into consideration of 40 project dataset. Similarly, Figures 3.5c and 3.5d display the actual eﬀort and the predicted eﬀort obtained for CP1 and CP2 respectively using the SGB technique taking into consideration of 30 project dataset. From this figure, it is observed that there is very little diﬀerence between the predicted eﬀort and the actual eﬀort.

#### Model Design using Various SVR Kernel Methods

#### After partitioning data into learning set and validation set, the model selection for *ϵ* and *γ* is performed using 5-fold cross validation process. In this study, in order to perform model selection, the *ϵ* and *γ* values are varied over a range. The *γ* value ranges from 2−7 to 27 and *ϵ* value ranges from 0 to 5. Hence, ninety models are generated to perform model selection operation.

Tables 3.8 and 3.9 show the validation error of ninety numbers of models generated for CP1 using SVR linear kernel and SVR polynomial kernel respectively based on the values of *ϵ* and *γ* for 40 project dataset. For SVR Linear kernel, *0.0065* value has been chosen as the minimum validation error. Hence based on the minimum.

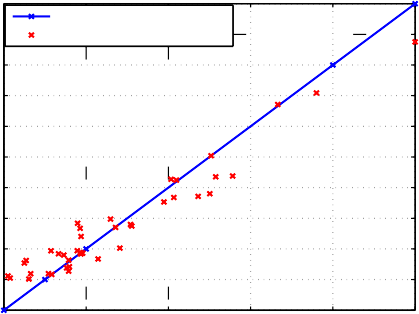
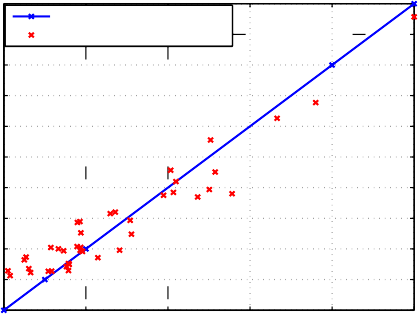
**Actual = Predicted**

**SGB CP2 Predicted Effort**

**0.8 0.8**

**0.7 0.7**

**Predicted Effort**



**0.6 0.6**

**Predicted Effort**

**0.5 0.5**

**0.4 0.4**

**0.3 0.3**

**0.2 0.2**

**0.1 0.1**

**0**

**0 0.2 0.4 0.6 0.8 1**

**Actual Effort**

(a)

**0**

**0 0.2 0.4 0.6 0.8 1**

**Actual Effort**

(b)

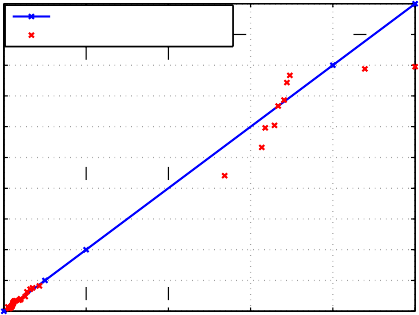
**Actual = Predicted**

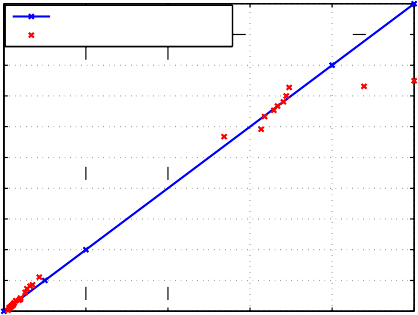
**SGB CP1 Predicted Effort**

**Actual = Predicted**

**SGB CP2 Predicted Effort**

**0.8 0.8**



**0.7 0.7**

**0.6 0.6**

**Predicted Effort**

**Predicted Effort**

**0.5 0.5**

**0.4 0.4**

**0.3 0.3**

**0.2 0.2**

**0.1 0.1**

**0**

**0 0.2 0.4 0.6 .8**

**Actual Effort**

(c)

**0.2 0.4 0.6 0.8 1**

**Actual Effort**

(d)

Figure 3.5: Actual vs. Predicted Eﬀort Graph using the SGB Technique for 40 and 30 Project Datasets

# CHAPTER 5

**USE CASE POINT APPROACH FOR SOFTWARE EFFORT ESTIMATION USING MACHINE LEARNING TECHNIQUES**

### Introduction

### Due to the increasing complexity of software development activities, the need for eﬀective eﬀort estimation techniques has arisen. Underestimation leads to disruption in the projects estimated cost and delivery. On the other hand, overestimation causes outbidding and financial losses in business. The job of software eﬀort estimation is a critical one in the early stages of the software development life cycle, when the details of requirements are usually not clearly identified. Hence, eﬀort estimation during early stage of software development life cycle (SDLC) plays a vital role for determining whether a project is feasible in terms of a cost-benefit analysis or not.

Use Case Point (UCP) approach relies on the use case diagram of UML paradigm for eﬀort estimation of a given software product. UCP helps in providing a comprehensive eﬀort estimation from the design phase itself. The total number of UCP is measured by ascertaining the total no. of use cases as well as actors and then multiplying each of them with their corresponding complexity factors. Each use case and actor are classified into one of the three classes such as simple, average and complex. The number of transactions per use case helps in determining its complexity value. The UCP model has broadly been utilized with in the most recent decade; still it possesses certain limitations. It assumes that the software size and eﬀort are directly proportional to each other. Due to this reason, the software eﬀort equation provided by UCP is not well accepted by software industries. Various optimization techniques help in improving the accuracy of eﬀort estimation. In this study, random forest and support vector regression techniques are employed to handle the confinements of the UCP model and to enhance prediction accuracy of software eﬀort estimation. The results obtained applying these techniques-based eﬀort estimation model is then measured against the results achieved applying Multi-Layer Perceptron (MLP), Radial Basis Function Network (RBFN), Stochastic Gradient Boosting (SGB) and Log-Linear Regression (LLR) models so as to critically evaluate their performances.

### Methodologies Used

### The methodologies that are utilized within this study to compute the eﬀort needed to create a software product are described below:

#### Use Case Point (UCP) Approach

The Use Case Point (UCP) approach was initially proposed by Gustav Karner in 1993. This approach is an extension of Function Point Approach (FPA) and Mk II FPA. If the idea about the problem domain, system size and architecture is known, then an early eﬀort estimation focused around use cases could be made. Figure 5.1 demonstrates the diﬀerent steps taken into consideration to compute the total no. of use case points.

**Final Use Case Point Evaluation**

**Calculation of TCF and EF**

**Calculation of Weights and Points**

**Classification of Actors and Use Cases**

**Use Case Diagram**

**Figure 5.1 Steps to Calculate Use Case Points**

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

##### Classification of Actors and Use Cases

|  |  |
| --- | --- |
| **Type of Actor** | **Corresponding Weight** |
| Simple | 1 |
| Average | 2 |
| Complex | 3 |

This step deals with classifying the actors in a use case diagram as simple, average or complex. An actor that represents a system with a well defined Application Programming Interface (API), is considered as simple. An actor that communicates with the system through a protocol, is classified as average. An actor is classified as complex, if it can represent a person who is interacting with system through a Webpage or Graphical User Interface (GUI). Each actor type is assigned with a weighting factor as shown in Table 5.1.

**Table 5.1 Assignment of Weighting Factors to Each Actor**

Similarly, a use case is classified into either simple or average or complex type. Classification of the use case depends on the number of transactions characterized in the description of use case along with secondary scenarios. A use case is considered *Simple*, if it uses less than four number of transactions to interact and also uses only a single database object. A use case is classified as *Average*, if it involves four to seven number of transactions and uses two or more database objects. A use case that involves more than seven number of transactions for processing and requires greater than or equal to three database objects, is considered as *Complex*. The complexity of a use case is characterized and weighted using the value given in Table 5.2.

|  |  |  |
| --- | --- | --- |
| **Type of Use Case** | **No. of Transactions** | **Corresponding Weight** |
| Simple | *<*= 3 | 5 |
| Average | 4 to 7 | 10 |
| Complex | *>*= 7 | 15 |

**Table 5.2: Assignment of Weighting Factors to Each Use Case**

##### Proposed Steps for Software Eﬀort Estimation

1. **Collection of Software Size, Productivity, Complexity and Actual Eﬀort Values**: The software size i.e., total number of use case points required to complete the project, productivity, complexity and actual eﬀort values for one hundred forty nine projects are collected from the literature.

|  |  |
| --- | --- |
| **Collection of Software Size, Productivity, Complexity and Actual Eﬀort Values** | |
|  |  |

**Data Normally**

**Distributed?**

no

yes

**Tranformation of Data**

|  |  |
| --- | --- |
| **Scaling of Data Set** | |
|  |  |
| **Division of Data Set** | |
|  |  |
| **Performing Model Selection** | |
|  |  |
| **Performance Evaluation** | |

**Figure 5.4 Proposed Steps for Software Eﬀort Estimation Purpose Applying RF and SVR**

Kernel Techniques RF model for calculating impact of each variable on the predicted eﬀort value. But after calculating the impact of each variable, the highest impact variable is considered as final input argument to the RF model.

1. **Data Normally Distributed**: The statistical analysis of the collected dataset has been performed. It is verified as to whether the collected dataset follows normal distribution or not, based on the values of skewness and kurtosis. If data are normally distributed, then it will directly proceed to the data normalization step. Otherwise, the data need to be transformed to make it more normally distributed.
2. **Transformation of Data**: If the dataset is not normally distributed, then the logarithmic transformation method has been applied over the dataset to make it normally distributed. Histograms have been plotted to properly verify the distribution of data before and after transformation.
3. **Scaling of Dataset** : The values taken as input arguments are individually scaled 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 a record ‘*s*’ is obtained by considering the following equation:

*Normalized*(*s*) = *s* − min(*S*)

max(*S*) − min(*S*)

(4.9)

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.

1. **Division of dataset**: Total no. of data are divided into two subsets i.e., training set and test set for both RF and SVR Kernel techniques. Random forest have randomness in input data and in splitting at nodes. Hence, in case of RF technique, initially an arbitrary random vector is selected to provide randomness in input data and to start the implementation process. Then, the data are divided using this arbitrary random vector.
2. **Performing Model Selection**: In case of RF technique, prediction results vary according to random vector. So an evaluation function, (1- MMER + 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. Then, by using this final random vector, results are being predicted.

Similarly, in case of SVR kernel-based eﬀort estimation model, the model which provides the least value than the other generated models based on the minimum validation error criteria has been selected to perform other operations. The tunable parameters have been selected to find the best parameter C and *γ* using a five-fold cross validation procedure . Based on the minimum validation error, the best model has been selected and the corresponding value of *γ* and *ϵ* value is found out. The final model selected based on best parameter of C, *ϵ* and *γ* has been trained using all training samples. The output of this step is the trained SVM model providing predicted response values for test inputs.

1. **Performance Evaluation**: In this study, the Mean Magnitude of Error Relative to the estimate (MMER) and the Prediction Accuracy (PRED(x)) are the two measures used to evaluate the performance of the model for test samples. Results obtained from proposed model-based on RF and SVR Kernel techniques are then evaluated against existing results to access its performance accuracy.

**CHAPTER 6**

**CONCLUTION**

It is observed in literature that analysts and practitioners have proposed several techniques for software eﬀort estimation purpose. However, the CP, UCP and SPA are one of the eﬀort estimation models which are used because of their simplicity, fastness and accuracy to a certain degree. The research contributions, conclusive remarks taking into account of the experimental research work carried out are incorporated into this chapter alongside the scope for future work.

**REFERENCE:**

* + 1. Robert N Charette. Why software fails. *IEEE spectrum*, 42(9):36, 2005.
    2. Sanchoy K Das, Pradeep Yedlarajiah, and Raj Narendra. An approach for estimating the end-of-life product disassembly eﬀort and cost. *International Journal of Production Research*, 38(3):657–673, 2000.
    3. STANDISH GROUP et al. Chaos manifesto. *The Standish Group*, 2013.
    4. Onur Demir¨ors and C¸ i˘gdem Gencel. A comparison of size estimation techniques applied early in the life cycle. In *Software Process Improvement*, pages 184–194. Springer, 2004.
    5. Ian Sommerville and Gerald Kotonya. *Requirements engineering: processes and techniques*. John Wiley & Sons, Inc., 1998.
    6. D Eck, B Brundick, T Fettig, J Dechoretz, and J Ugljesa. Parametric estimating handbook.

*The International Society of Parametric Analysis (ISPA),*, 2009.

* + 1. Ali Bou Nassif. *Software Size and Eﬀort Estimation from Use Case Diagrams Using Regression and Soft Computing Models*. PhD thesis, Western University, 2012.
    2. STANDISH GROUP et al. The chaos manifesto 2011. *The Standish Group International. EUA*, 2011.
    3. Donald J Reifer. Web development: estimating quick-to-market software. *IEEE software*, 17(6):57–64, 2000.
    4. ISBSG. The international software benchmarking standards group. [http://www.isbsg.org,](http://www.isbsg.org/) 2011.
    5. Ofer Morgenshtern, Tzvi Raz, and Dov Dvir. Factors aﬀecting duration and eﬀort estimation errors in software development projects. *Information and Software Technology*, 49(8):827–837, 2007.
    6. Adriano LI Oliveira. Estimation of software project eﬀort with support vector regression.

*Neurocomputing*, 69(13):1749–1753, 2006.

* + 1. Anna Corazza, Sergio Di Martino, Filomena Ferrucci, Carmine Gravino, Federica Sarro, and Emilia Mendes. How eﬀective is tabu search to configure support vector regression for eﬀort estimation? In *Proceedings of the 6th international conference on predictive models in software engineering*, page 4. ACM, ACM, 2010.
    2. Anna Corazza, Sergio Di Martino, Filomena Ferrucci, Carmine Gravino, and Emilia Mendes. Investigating the use of support vector regression for web eﬀort estimation. *Empirical Software Engineering*, 16(2):211–243, 2011.
    3. Petrˆonio L Braga, Adriano LI Oliveira, and Silvio RL Meira. A ga-based feature selection and parameters optimization for support vector regression applied to software eﬀort estimation. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1788–1792. ACM, 2008.
    4. James N Morgan and John A Sonquist. Problems in the analysis of survey data, and a proposal.

*Journal of the American statistical association*, 58(302):415–434, 1963.

APPENDIX - A

from \_\_future\_\_ import print\_function, division

import sys

import os

sys.path.append(os.path.abspath("."))

sys.dont\_write\_bytecode = True

# from datasets.albrecht import Albrecht

# from datasets.china import China

# from datasets.desharnais import Desharnais

# from datasets.finnish import Finnish

# from datasets.isbsg10 import ISBSG10

# from datasets.kemerer import Kemerer

# from datasets.kitchenhamm import Kitchenhamm

# from datasets.maxwell import Maxwell

# from datasets.miyazaki import Miyazaki

from datasets.cleaned.albrecht import Albrecht

from datasets.cleaned.china import China

from datasets.cleaned.desharnais import Desharnais

from datasets.cleaned.finnish import Finnish

from datasets.cleaned.isbsg10 import ISBSG10

from datasets.cleaned.kemerer import Kemerer

from datasets.cleaned.kitchenhamm import Kitchenhamm

from datasets.cleaned.maxwell import Maxwell

from datasets.cleaned.miyazaki import Miyazaki

from utils.lib import \*

from utils.validation import \*

from methods.peeking import peeking2

from methods.cart import cart

from methods.teak import teak

from methods.knn import knn\_1, knn\_3

from methods.cogee import cogee

from methods.atlm import atlm

from optimizer.teak\_optimize import teak\_optimize

from utils.errors import \*

from utils import sk

from joblib import Parallel, delayed

from time import time

datasets = [Albrecht, Desharnais, Finnish, Kemerer, Maxwell,

Miyazaki, China, ISBSG10, Kitchenhamm]

error = msae

def mre\_calc(y\_predict, y\_actual):

mre = []

for predict, actual in zip(y\_predict, y\_actual):

mre.append(abs(predict - actual) / (actual))

mmre = np.median(mre)

if mmre == 0:

mmre = np.mean(mre)

return mmre

def sa\_calc(y\_predict, y\_actual):

ar = 0

for predict, actual in zip(y\_predict, y\_actual):

ar += abs(predict - actual)

mar = ar / (len(y\_predict))

marr = sum(y\_actual) / len(y\_actual)

sa\_error = (1 - mar / marr)

return sa\_error

def run(reps=1):

for dataset\_class in datasets:

dataset = dataset\_class()

model\_scores = {"CART": N(),

"PEEKING": N(),

"TEAK": N(),

"KNN1": N(),

"KNN3": N(),

"ATLM": N(),

"COGEE": N(),

"O\_TEAK": N()

}

for score in model\_scores.values():

score.go = True

for \_ in xrange(reps):

for test, rest in kfold(dataset.get\_rows(), 3, shuffle=True):

say(".")

desired\_effort = [dataset.effort(row) for row in test]

all\_efforts = [dataset.effort(one) for one in rest]

model\_scores["PEEKING"] += error(desired\_effort, peeking2(dataset, test, rest), all\_efforts)

model\_scores["CART"] += error(desired\_effort, cart(dataset, test, rest), all\_efforts)

model\_scores["TEAK"] += error(desired\_effort, teak(dataset, test, rest), all\_efforts)

model\_scores["KNN1"] += error(desired\_effort, knn\_1(dataset, test, rest), all\_efforts)

model\_scores["KNN3"] += error(desired\_effort, knn\_3(dataset, test, rest), all\_efforts)

model\_scores["ATLM"] += error(desired\_effort, atlm(dataset, test, rest), all\_efforts)

model\_scores["COGEE"] += error(desired\_effort, cogee(dataset, test, rest), all\_efforts)

model\_scores["O\_TEAK"] += error(desired\_effort, teak\_optimize(dataset, test, rest), all\_efforts)

sk\_data = [[key] + n.cache.all for key, n in model\_scores.items()]

print("\n### %s (%d projects, %d decisions)" %

(dataset\_class.\_\_name\_\_, len(dataset.get\_rows()), len(dataset.dec\_meta)))

print("```")

sk.rdivDemo(sk\_data)

print("```")

print("")

def run\_for\_dataset(dataset\_class, dataset\_id, reps):

write\_file = "results/%s\_sa\_mre.txt" % dataset\_class.\_\_name\_\_

with open(write\_file, "wb") as f:

dataset = dataset\_class()

dataset\_name = dataset\_class.\_\_name\_\_

print("\n### %s (%d projects, %d decisions)" %

(dataset\_name, len(dataset.get\_rows()), len(dataset.dec\_meta)))

# folds = 3 if len(dataset.get\_rows()) < 40 else 10

folds = 3

for rep in range(reps):

fold\_id = 0

for test, rest in kfold(dataset.get\_rows(), folds, shuffle=True):

print("Running for %s, rep = %d, fold = %d" % (dataset\_name, rep + 1, fold\_id))

fold\_id += 1

all\_efforts = [dataset.effort(one) for one in rest]

actual\_efforts = [dataset.effort(row) for row in test]

start = time()

atlm\_efforts = atlm(dataset, test, rest)

atlm\_end = time()

cart\_efforts = cart(dataset, test, rest)

cart\_end = time()

cogee\_efforts = cogee(dataset, test, rest)

cogee\_end = time()

atlm\_mre, atlm\_sa = mre\_calc(atlm\_efforts, actual\_efforts), msa(actual\_efforts, atlm\_efforts, all\_efforts)

cart\_mre, cart\_sa = mre\_calc(cart\_efforts, actual\_efforts), msa(actual\_efforts, cart\_efforts, all\_efforts)

cogee\_mre, cogee\_sa = mre\_calc(cogee\_efforts, actual\_efforts), msa(actual\_efforts, cogee\_efforts, all\_efforts)

f.write("%s;%d;%f;%f;%f\n" % (dataset\_name, 1, atlm\_mre, atlm\_sa, atlm\_end - start))

f.write("%s;%d;%f;%f;%f\n" % (dataset\_name, 2, cart\_mre, cart\_sa, cart\_end - start))

f.write("%s;%d;%f;%f;%f\n" % (dataset\_name, 3, cogee\_mre, cogee\_sa, cogee\_end - start))

return write\_file

def run\_patrick(reps, num\_cores, consolidated\_file="results/patrick\_sa\_mre.txt"):

local\_datasets = datasets

# local\_datasets = [Miyazaki]

dataset\_files = Parallel(n\_jobs=num\_cores)(delayed(run\_for\_dataset)(dataset\_class, dataset\_id, reps)

for dataset\_id, dataset\_class in enumerate(local\_datasets))

with open(consolidated\_file, "wb") as f:

f.write("dataset;method;SA;MRE;Runtime\n")

for dataset\_file in dataset\_files:

with open(dataset\_file) as df:

for line in df.readlines():

if len(line) > 0:

f.write("%s" % line)

# os.remove(dataset\_file)

def sarro\_cogee\_dataset(dataset\_class, error, folds, reps):

dataset = dataset\_class()

print("\n### %s (%d projects, %d decisions)" %

(dataset\_class.\_\_name\_\_, len(dataset.get\_rows()), len(dataset.dec\_meta)))

model\_scores = {"CART": N(),

"ATLM": N(),

"COGEE": N()

}

for score in model\_scores.values():

score.go = True

for \_ in range(reps):

for test, rest in kfold(dataset.get\_rows(), folds, shuffle=True):

say(".")

desired\_effort = [dataset.effort(row) for row in test]

all\_efforts = [dataset.effort(one) for one in rest]

model\_scores["CART"] += error(desired\_effort, cart(dataset, test, rest), all\_efforts)

model\_scores["ATLM"] += error(desired\_effort, atlm(dataset, test, rest), all\_efforts)

model\_scores["COGEE"] += error(desired\_effort, cogee(dataset, test, rest), all\_efforts)

sk\_data = [[key] + n.cache.all for key, n in model\_scores.items()]

print("```")

stat = sk.rdivDemo(sk\_data)

print("```")

print("")

write\_file = "%s/%s.txt" % ("results/sarro", dataset\_class.\_\_name\_\_)

with open(write\_file, "wb") as f:

f.write("\n### %s (%d projects, %d decisions)\n" %

(dataset\_class.\_\_name\_\_, len(dataset.get\_rows()), len(dataset.dec\_meta)))

f.write("```\n%s\n```\n\n" % stat)

return write\_file

def sarro\_cogee(num\_cores, folds=3, reps=10):

datasets = [China, Desharnais, Finnish, Maxwell, Miyazaki,

Albrecht, Kemerer, ISBSG10, Kitchenhamm]

# datasets = [Miyazaki, Finnish]

mkdir("results/sarro")

error = msa

dataset\_files = Parallel(n\_jobs=num\_cores)(delayed(sarro\_cogee\_dataset)(dataset\_class, error, folds, reps)

for dataset\_id, dataset\_class in enumerate(datasets))

consolidated\_file = "results/sarro/sa.md"

with open(consolidated\_file, "wb") as f:

for dataset\_file in dataset\_files:

with open(dataset\_file) as df:

for line in df.readlines():

f.write(line)

def run\_patrick\_v2():

reps = 20

folds = 3

contents = []

for dataset\_class in datasets:

dataset = dataset\_class()

dataset\_name = dataset\_class.\_\_name\_\_

start = time()

atlm\_mres, atlm\_sas = [], []

for rep in range(reps):

fold\_id = 0

for test, rest in kfold(dataset.get\_rows(), folds, shuffle=True):

print("Running for %s, rep = %d, fold = %d" % (dataset\_name, rep + 1, fold\_id))

fold\_id += 1

all\_efforts = [dataset.effort(one) for one in rest]

actual\_efforts = [dataset.effort(row) for row in test]

atlm\_efforts = atlm(dataset, test, rest)

atlm\_mre, atlm\_sa = mre\_calc(atlm\_efforts, actual\_efforts), msa(actual\_efforts, atlm\_efforts, all\_efforts)

atlm\_mres.append(atlm\_mre)

atlm\_sas.append(atlm\_sa)

end = time()

content = "dataset: %s\ntotal runtime: %f\n" % (dataset\_name, end - start)

content += "\nMRE\n" + " ".join(map(str, atlm\_mres)) + "\n"

content += "\nSA\n" + " ".join(map(str, atlm\_sas)) + "\n"

contents.append(content)

with open("results/patrick\_sa\_mre\_v2.txt", "wb") as f:

f.write("\n########################################\n\n".join(contents))

def \_sarro():

reps = 10

folds = 3

cores = 16

sarro\_cogee(cores, folds, reps)

def \_main():

# reps = 20

# cores = 16

# consolidated\_file = "results/patrick\_sa\_mre\_v2.txt"

# run\_patrick(reps, cores, consolidated\_file)

run\_patrick\_v2()

if \_\_name\_\_ == "\_\_main\_\_":

\_main()

# \_sarro()