Dayananda Sagar University
Department of Computer Science & Engineering
School of Engineering
Dayananda Sagar University
Kudlu Gate, Bangalore - 560068

Major Project Report (Phase - 1)

on

Software Effort Estimation Using Machine Learning Techniques

VII Semester
Course Code: (16CS481)

Bachelor of Technology
in
Computer Science & Engineering

Submitted by:

B P Gayathri Ananya (ENG17CS0047)
Bharat Nilam (ENG17CS0050)
Chirag P D (ENG17CS0059)

Under the guidance of
Dr. Shyamsundar Pandeya

Dayananda Sagar University

School of Engineering, Kudlu Gate, Bangalore - 560068

CERTIFICATE

This is to certify that B P Gayathri Ananya, Bharat Nilam and Chirag P D bearing USNs ENG17CS0047, ENG17CS0050 and ENG17CS0059 has satisfactorily completed his/her Major Project (Phase - 1) as prescribed by the University for the 7th Semester B.Tech programme in Computer Science & Engineering for the Major Project (16CS481) course during the year 2020 at the School of Engineering, Dayananda Sagar University, Bangalore.

Date: Signature of Supervisor

Max Marks	Marks Obtained

Signature of Chairman
Department of Computer Science & Engineering

ACKNOWLEDGMENT

From the very core of our heart, we would like to express our sincere gratitude to **Dr. Shyamsundar Pandeya** for his invaluable guidance, support, motivation and patience during the course of this major project work. We are always indebted to him for his kind support and constant encouragement.

We extend our sincere thanks to our **Chairman Dr. Sanjay Chitnis** who continuously helped throughout the project and without his guidance, this project would have been an uphill task.

It requires lots of efforts in terms of cooperation and support to fulfill various tasks involved during the project. We are always grateful to our peers and friends who have always encouraged us and guided us whenever we needed assistance.

B P Gayathri Ananya (ENG17CS0047)
Bharat Nilam (ENG17CS0050)
Chirag P D (ENG17CS0059)

Contents

Abstract

In software engineering, the main aim is to develop a high-quality project that fall within scheduled time and budget, this procedure is called effort estimation. Effort estimation is crucial and important for a company to do because hiring more people than needed will lead to loss of income, and hiring less people than needed will lead to delay of project delivery. The aim of this study is to estimate software effort objectively by using machine learning techniques instead of subjective and time-consuming estimation methods. We would be using decision tree. We are using the boosting algorithm to increase the accuracy level of our ensemble model which is a combination of SVM, decision tree and GLM, ensemble learning will be tried on two public datasets namely Desharnais and Maxwell.

1 Introduction

Successful project is that the system is delivered on time and within budget and with the required quality.

In software development, effort estimation is the process of predicting the most realistic amount of effort (expressed in terms of person-hours or money) required to develop or maintain software based on incomplete, uncertain and noisy input.

Software researchers and practitioners have been addressing the problems of effort estimation for software development projects since at least the 1960s.

Most of the research has focused on the construction of formal software effort estimation models. The early models were typically based on regression analysis or mathematically derived from theories from other domains. Since then a high number of model building approaches have been evaluated, such as approaches founded on case-based reasoning, classification and regression trees, neural networks, genetic programming etc.

The most common estimation methods today are the parametric estimation models COCOMO, SEER-SEM and SLIM.

The product/software effort/cost-estimation techniques are applied to predict the effort required to finish the project. An incorrect estimation leads to increase in deadline and budget of the project which may further consequence to failure of the project.

Effort estimation is crucial and important for a company to do because hiring more people than needed will lead to loss of income, and hiring less people than needed will lead to delay of project delivery.

The estimation models and techniques are used in different phases of software engineering like budgeting, risk analysis, planning, etc.

2 Problem Statement

Develop an effective effort estimation model achieving best possible accuracy level, optimizing software projects by estimating efforts for the same using machine learning techniques.

3 Literature Survey

Omar Hidmi et al. [1] aim to estimate software effort objectively by using machine learning techniques instead of using subjective and time consuming estimation methods like expert judgment and estimation by analogy. To add more, unsuitable criteria and technique for estimation may be chosen by an expert. For these reasons, it is highly advantageous to use a more structured estimation process using machine learning techniques.

Data and methods:

- In this research, they have used two publicly available datasets, desharnais and maxwell , to build a model for estimating the effort for new software development projects. They are two of the most commonly used datasets in the field of software effort estimation.

- Desharnais dataset consists of 81 projects collected by J.M. desharnais in the late 1980s from a Canadian software house.

- Maxwell dataset is a relatively new dataset consists of 62 projects between 1985 and 1993 [7]. Each project is described by 27 attributes in which all attributes are numerical.
- The two mentioned datasets have the same measure type of effort which is person-hours

Machine Learning techniques use:

- To form training set and testing set, we randomly divided the datasets using two techniques which are leave-one-out cross validation and k-fold cross-validation. The two datasets were analysed in their own context by using two machine learning techniques which are k-nearest neighbour (k-nn) and support vector machine (svm).
- The accuracy of the model is increased by using the adaptive boost algorithm.

Results: when applying a single method alone, it has a good accuracy equals to 85% in the best scenario (this was a result when applying svm technique using 2 classes to desharnais dataset), but when we combined the classifiers, we get 91.35% accuracy when using desharnais dataset and 85.48% accuracy when using maxwell dataset. So, we can say that boosting one technique with another improves the accuracy of estimations.

Pospieszny P et al. [**2**] proposed an effort and duration estimation model using smart data preparation, a set of three machine algorithm (named as support vector machine, multi-layer perceptron and generalized linear models) and validated using three-fold cross-validation, whose purpose is to work as a decision support tool for any organization. The authors depicted that this model is suitable for medium and large scale organization which have significant volume of finished projects. The proposed model is more appropriate in the initial stages of software development life cycle where uncertainty of deliverable product is high. The authors concluded that the results imply very good prediction accuracy.

Qi F, Jing XY et al. [**3**] did a study on available data sets for software effort estimation and found that most of the organizations did not share the project's effort data because of privacy concern which lead a limited amount of effort data. Software effort estimation on limited data gives an unrealistic estimation. Due to this, the authors proposed a method to reduce the shortage of data by choosing GitHub data and also proposed AdaBoost and Classification and Regression Tree (ABCART), a sample incremental algorithm that increases the samples of collected data sets online and also satisfy the requirement of the ever-changing growth of data sets. Authors concluded that effort estimation for new projects that do not have training data can be easily done by using open source project (OSP) data. Experimental result disclosed that the estimation performed on collected data from OSP has comparable performance with those of existing effort data sets.

Petrônio L. Braga et al. [**4**] propose the use of robust confidence intervals for defining a confidence interval for software effort estimates.

- Robust confidence intervals are directly computed from the prediction errors in the training set of the regression algorithm for the datasets under analysis.

- A robust confidence interval for predictions is computed from errors collected from the training set after the collection of errors used to calculated Sn(e) is representative of the errors that will be obtained when the regression model is used in practice.

- If the collection of errors is large enough then Sn(e) can be assumed to be close to Fn(e), the true error distribution. In this case the confidence intervals for upcoming predictions are computed simply by keeping as much of the interior of Sn(e) as is desired and by using the limits of this truncated collection to define the confidence interval.

Let n be the number of training samples and p be the desired confidence level.
For very large error collections and moderate values of p, n x p values should be discarded from each extreme in order to build the confidence intervals. For smaller samples, however, the recommended amount to be discarded from each extreme is n x p - 1 [10]. If the result is a real number it should be truncated. p is the fraction of probability in each tail of the distribution function.
The process used to build robust confidence intervals can be divided in six stages, described below:

1. Train a regression model using n projects;

2. Obtain the collection of n errors;

3. Sort the collection of errors in ascending order;

4. Calculate the fraction of probability;

5. Discard (n x p – 1) errors from each extreme of the collection of errors;

6. Obtain robust confidence intervals for future predictions.

4 Requirement Analysis

4.1 Gathering Datasets

4.1.1 Desharnais Dataset

Desharnais dataset consists of 81 projects collected by J.M. Desharnais in the late 1980s from a Canadian software house. The original dataset consists of 12 attributes: Team experience, Manager experience, year project end, entities, transactions, length, points non adjust, points adjust, adjustment, effort(dependent), project ID and language. Despite the fact that this dataset is now more than 25 years old, it is one of the largest and most used publicly available datasets.

4.1.2 Maxwell Dataset

Maxwell dataset is a relatively new dataset consists of 62 projects between 1985 and 1993. Each project is described by 27 attributes in which all attributes are numerical.

The attributes are software year, application type, hardware platform, database, user interface, source, telon use, number of languages used, customer participation, Development Environment, Staff Availability Standards Use, Methods Use, Tools Use, Software's Logical Complexity, Requirements Volatility, Quality Requirements, Efficiency Requirements, Installation Requirements, Staff Analysis Skills, Staff Application Knowledge, Staff Tool Skills, Staff Team Skills Duration (months), Application Size (Function Points), Time and effort.

4.2 Data Processing

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues.

The steps involved in data processing are:

- Acquiring the dataset and importing all the required libraries for data processing:
  - The three main libraries that will be used are:
    - NumPy
    - Pandas
    - Matplotlib
- Identifying and handling the missing values: This step involves filling in the missing values with appropriate data like the mean of that attribute to get accurate results.

Software Requirements

- Anaconda Environment
- Jupyter Notebook
- Python libraries such as NumPy, pandas, Matplotlib, etc
- Operating system: Windows 8 or newer, 64-bit macOS 10.13+, or Linux, including Ubuntu, RedHat, CentOS 6+, and others

Hardware Requirements

- Physical server or virtual machine
- System architecture: Windows- 64-bit x86, 32-bit x86; MacOS- 64-bit x86; Linux- 64-bit x86
- Minimum 5 GB disk space

5 Design

5.1 Encoding the categorical data

Machine Learning models are primarily based on mathematical equations. Thus, we can intuitively understand that keeping the categorical data in the equation will cause certain issues since you would only need numbers in the equations. So we must convert it into numerical values. To do so, we are using the LabelEncoder() class from the sci-kit learn library.

5.2 Researching for the best technique

The techniques that we will use for this project are:

- SVM - Support Vector Regression
- Decision Tree
- GLM - General Linear Model

5.3 Training and Testing Data

To create 80% training data set and 20% testing data sets we will split the dataset using K-Fold Cross Validation.

In this method, we split the data-set into k number of subsets(known as folds) then we perform training on the all the subsets but leave one(k-1) subset for the evaluation of the trained model. In this method, we iterate k times with a different subset reserved for testing purpose each time.

5.4 Evaluation

The metrics used for evaluation are:

- Classification metrics
- Mean squared error

6 Conclusion

From the above research papers that we have put up, we tried to analyze multiple perspectives to optimize the outcome with minimal effort. The key point here is the accuracy level each method presents, as seen in the Software effort estimation based on open source projects: Case study of GitHub paper, the accuracy level is ranged from 70%-80%. The ensemble approach using the AdaBoost algorithm over k-nn and SVM methods gave an accuracy close 91% to which was comparatively more appealing. We want to go ahead by using a boosting method to obtain higher accuracy level and also try to predict the confidence intervals.

7 Future Work

Some limitations in this domain are:

- Estimation of time and effort in earlier phase of software development is very difficult and it depends on lower level of estimation such as Size Estimation which is done by using External Inputs (EI), External Outputs (EO), External Queries (EQ), Internal Logical Files (ILF), and External Interface Files (EIF).
- Many existing research papers have proposed various effort estimation techniques and they still do not have an agreement which technique is the best across different cases.
- Also we don't have any dynamic learning algorithm for our model to adopt itself with any situation and completed our database in each estimation time. By adding the process maturity in effort estimation models as an input factor, we can improve the accuracy of estimation models.

This limitation gives us motivation to continue this research in our future work.

References

[1] Omar Hidmi, and Betul Erdogdu Sakar (2017) Software Development Effort Estimation Using Ensemble Machine Learning. Journal: Int'l Journal of Computing ISSN 2349-1469 EISSN 2349-1477

[2] Pospieszny P, Chrobot BC, Kobyli´nski A (2017) An effective approach for software project effort and duration estimation with machine learning algorithms. Journal: J Syst Softw 137:184–196

[3] Qi F, Jing XY, Zhu X, Xie X, Xu B, Ying S (2017) Software effort estimation based on open source projects: case study of GitHub. Journal: Inf Softw Technol 92:147–157

[4] Petrônio L. Braga and Adriano L. I. Oliveira (2007) Software Effort Estimation using Machine Learning Techniques with Robust Confidence Intervals. Conference: Hybrid Intelligent Systems, DOI: 10.1109/HIS.2007.56