Software effort estimation with Machine Learning

Thi Hong Tran, Åke Tyvi, Kristian Eriksso

Section I. Introduction

Software Effort Estimation is essential to any software house, institution or governmental bureau having large projects, using budgeting and having limited funding. It can be used for predicting work effort estimation, project cost, monitoring project progress and recounting final cost estimation while the project is still running.
Building Software has moved during the years from Royce's Waterfall Model into Agile Methodology. During this time, there have been several Software Development Methodologies involved in producing the software: For example Spiral, Chaos and Prototyping were common methods of the past. During the development of Software Development Methodologies, several methods predicting Software Effort and Cost were also developed: Direct curve fitting, Cocomo '81, Cocomo II, Function Point Analysis or Expert Judgement to mention just a few.
In the past, Software Costs were mainly estimated beforehand, while in these days software is developed until a client stops the development (i.e. either the product is ripe enough, or the project runs out of money). Even so, the fundamental principles of Putnam's and Royce's Cost Division model can also be found hidden inside Agile's sprint (as can also be shown according to the Lawrence H Putnam's paper ‘'A General Empirical Solution to the Macro Software Sizing and Estimating Problem", where paper is referring to the fact of Software Development as 'the laws of physics related in software development' costs). We may argue about these existing labor cost division attributes forming part of the Effort Costs, but they are an existing fact. However, the Software Cost Effort Estimation process is more complex than just the basic workflow process division, as those having constructed formal models did notice. Several attributes contribute to the Effort Estimation, and even so, it seems to be difficult, or impossible, to make accurate enough predictions or results comparable between organizations. For this reason, applying Machine Learning (ML) and Artificial Intelligence (AI) solutions in the field of Software Cost Estimation has gained popularity; modern ML and AI solutions give more accurate predictions to formal methods' predictions.

Various machine learning models have been developed to use data from past projects to build estimation models. In this project, COCOMO'81 and COCOMO II are selected to do experiments between multiple regression methods and ensemble methods (boosting, bagging). The aim is to find out which methods are applicable to be used in software cost estimation.

This report is organized as follows. In Section II, the used Machine Learning methods and evaluation criteria are explained in depth. The experiments and results are discussed in Section III. Finally, Section IV is our conclusion.

Section II. Mathematical modeling

2.1. Regression models

Regression algorithms predict the output values (as a continuous numerical value) based on the input features from the data fed in the system. Regression analysis finds out the relationship between a single dependent variable (target variable) and several independent ones.

- **Decision Tree and Random Forest**

The decision tree models can be applied to all data which contain numerical features and categorical features. They are good at capturing non-linear interaction between the features and the target variable. But they are prone to overfitting, especially when a tree is particularly deep. Random forests are a strong modeling technique and much more robust than a single decision tree. They aggregate many decision trees to limit overfitting as well as error due to bias and therefore yield useful results [1].

- **Support Machine Regression**

Support Vector Regression uses the same idea of SVM, but was designed for regression problems. SVR tries to fit the error within a certain threshold instead of minimizing the error rate as a simple regression model does [2].

- **LASSO and Ridge Regression**

Regularization is an approach that prefers some bias over high variance. Ridge regression uses L2 regularization techniques. LASSO regression uses L1 regularization techniques. While Ridge adds the squared value of the coefficients in the cost function, LASSO adds the penalty term in the cost function by adding the absolute value of the coefficients [3].
LASSO avoids overfitting problems and is helpful when the number of feature points are large in number. Unlike LASSO, Ridge regression is not good at feature reduction since it never leads to a coefficient being zero rather only minimizes it.

- **Ensemble techniques**

Ensemble learning is a machine learning paradigm where multiple models (often called “weak learner”) are trained to solve the same problem and combined to get better results. A low bias and a low variance, although they most often vary in opposite directions, are the two most fundamental features expected for a model. The idea of ensemble methods is to try reducing bias and/or variance of such weak learners by combining several of them together in order to create a strong learner (or ensemble model) that achieves better performances. There are three major kinds of meta-algorithms that aim at combining weak learners: bagging, boosting, and stacking. Bagging will mainly focus on getting an ensemble model with less variance than its components whereas boosting and stacking will mainly try to produce strong models less biased than their components (even if variance can also be reduced) [4].
In this project, we carried out experiments on Random Forest (an improvement of Bagging) and AdaBoost (adaptive boosting).

2.2. Evaluation criteria

The following evaluation metrics are adapted to assess and evaluate the performance of the cost estimation models [5].

- **Magnitude of Relative Error (MRE)**: calculate the degree of estimation error for individual project

$$MRE_i = \frac{\left|Actual\ Cost_i - Estimated\ Cost_i\right|}{Actual\ Cost_i} \times 100$$

- **Mean Magnitude of Relative Error (MMRE)**: calculate the percentage of absolute values of relative errors.

$$MMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|Actual\ Cost_i - Estimated\ Cost_i\right|}{Actual\ Cost_i}$$

- **Median Magnitude of Relative Error (MdMRE)**: find the median of absolute values of relative errors.

$$MdMRE = median(MRE_i)$$

- **Prediction Level (PRED)**:
  Prediction at level n is defined as the % of projects that have absolute relative error less than n. Another widely used prediction quality indicator is pred(l), which is simply the percentage of estimates that are within the actual value.
  Conte, Dunsmore, and Shen suggest that an acceptable value of MRE is a value less or equal 0.25.

$$PRED(l) = \frac{k}{n} \times 100$$

  Where l is the maximum MRE of a selected range, n is the total number of projects, and k is number of projects in a set of n projects whose $MRE \leq l$

Section III. Experiment

3.1. Scenarios

In this project, we used two software cost datasets to evaluate the models using Regression algorithms. The first was the Cocomo81 dataset. The second was a dataset from NASA. The cocomo81 dataset consists of 17 attributes and 63 records. The Cocomo NASA 2 dataset consists of 24 attributes and 93 records. The output attributes of both datasets are in the units of person-months. Some of the columns of the NASA dataset are categorical, so we transformed the categorical values into their numeric representation. We also dropped the 'recordnumber' column before the experiment.

Dataset Name	No. of Records	No. of Attributes	Output Attribute-Effort (unit)
Cocomo81 [ref]	63	17	Person-months
Cocomo NASA 2 [ref]	93	24	Person-months

To form the test set, we randomly selected 25% data. The 10 folds' cross-validation technique was used to train the remaining data of each dataset.

Single models considered for comparison were SVM, Decision Tree, Ridge, LASSO. Selected ensemble techniques were bagging (Random Forest - an improvement over bagged decision trees), boosting (AdaBoost).
When we tuned the hyperparameters, there were a large number of parameters having many possible values like Decision Tree or Random Forest. It might be a costly and time-consuming task, so Randomized Search was our selection. Instead of searching for all combinations of hyperparameters, it samples a fixed number of parameter settings from the specified distributions, and saves computation resources [6].
To evaluate the models and measure the accuracy, three metrics have been used; MMRE, MdMRE, and PRED(0.25).

3.2. Results and discussion

For experimental analysis, we have chosen randomly 25% projects from each dataset.
The table below is the results when estimating on projects from the Cocomo81 dataset by 6 algorithms.

Criteria / Algorithms	SVR	LASSO	Ridge	AdaBoost	Random Forest	Decision Tree
MMRE	1.222	44.5	5.752	2.738	1.975	31.943
MdMRE	0.657	19.068	2.816	0.956	1.314	12.730
PRED(.25)	25.0	12.5	6.25	12.5	6.25	0.0

The table below is the results when estimating on projects from the Cocomo NASA 2 dataset by 6 algorithms.

Criteria / Algorithms	SVR	LASSO	Ridge	AdaBoost	Random Forest	Decision Tree
MMRE	1.456	2.461	1.965	1.629	0.900	1.296
MdMRE	0.654	0.895	0.802	0.753	0.553	0.771
PRED(.25)	16.667	8.333	20.833	25.0	37.5	16.667

We can see that 6 methods performed differently on each dataset.
On Cocomo81, SVR had the best performance on Cocomo81 with the lowest MMRE, MdMRE and the highest PRED(.25). There were no projects that the estimation made by Decision Tree models was within 25% actual cost, but this statistic is acceptable when the MMRE and MdMRE were so high, in second place. Meanwhile, LASSO made 12.5 at PRED(.25) with the highest MMRE and MdMRE (44.5 and 19.068 respectively); in other words, LASSO was the worst model. Ridge, AdaBoost and Random Forest had moderate efficiency.
On Cocomo NASA 2, the overall performance was improved. LASSO still did the worst, the biggest MMRE, MdMRE and the lowest PRED(.25) (approximately 8.33). Random Forest - the lead of 6 algorithms, had the smallest MMRE and MdMRE, the biggest PRED(.25) of

37.5. AdaBoost was the following model, with PRED(.25) of 25.0. Two ensemble methods were doing good estimation on the NASA 2 dataset. With the same performance at PRED(.25), nearly 16.67, SVR was considered to be better than Decision Tree in comparison of MMRE and MdMRE.
To sum up, ensemble methods (AdaBoost, Random Forest) did well on both datasets.

3.3. Deliverables

With the aim of having a general perspective of software cost estimation attributes and the way the COCOMO model works, we built a cocomo-app, where people could try estimating their own projects' effort and development time. We used the COCOMO model [7] as the core algorithm behind.

Section IV. Conclusion

In the proposal, we planned to do experiments on different types of machine learning and also Neural Networks. However, the output of a software cost estimation is a continuous numerical value, regression techniques generally outperform in this type of problem, so we decided to concentrate on Regression and Ensemble methods so as to have a deep understanding of them and to assess their actual performance on a specific domain.
Software effort estimation is totally new for us, with multiple underlying attributes and its own evaluation criteria: MMRE, MdMRE and PRED(.25). It's good to know that a good evaluation model has the high PRED(.25) and the low MMRE or MdMRE.
Firstly, to create a good model, the way of splitting a dataset needs to be considered carefully. K-fold cross-validation is a method to estimate the skill of a machine learning model on unseen data, resulting in a less biased or less optimistic estimate of the model skill than other methods (train/test split). This technique is integrated with Randomized Search (Grid Search also). Secondly, choosing hyperparameters and their range of values is also a difficult task. Too many hyperparameters or too narrow or too wide a range of values can result in ineffective tuning. The number of hyperparameters and their testing values should be also limited despite the computation support of Randomized Search. Finally, we had a chance to approach ensemble methods which are not strange, but we hadn't dug up them before. Different base-learner combinations can enhance the efficiency in different ways, as in the report, AdaBoost and Random Forest made better performance on both datasets than Decision Tree.
All deliverables of this project were published on Github.

References

1. Neil Liberman, Decision Trees and Random Forests, https://towardsdatascience.com/decision-trees-and-random-forests-df0c3123f991, last visited Oct 28, 2021.
2. Harris Drucker, Christ J.C. Burges, Linda Kaufman, Alex Smola and Vladimir Vapnik, N. (1997); "Support Vector Regression Machines", in *Advances in Neural Information Processing Systems 9, NIPS 1996*, 155–161, MIT Press.

3. Dhaval Taunk, L1 vs L2 Regularization: The intuitive difference, https://medium.com/analytics-vidhya/l1-vs-l2-regularization-which-is-better-d01068e6658c, last visited Oct 28, 2021.
4. Joseph Rocca, Ensemble methods: bagging, boosting and stacking, https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f, last visited Oct 28, 2021.
5. Shiyna Kumar, Vinay Chopra, "To Design and Implement Neural Network and Fuzzy Logic for Software Development Effort Prediction", International Journal of Computer Application (0975 - 8887), Vol 84, No.11, December 2013.
6. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html, last visited Oct 31, 2021.
7. https://en.wikipedia.org/wiki/COCOMO, last visited Oct 31, 2021.