International Journal of Innovations & Research Analysis (IJIRA)
ISSN :2583-0295, Impact Factor: 5.449, Volume 02, No. 02 (II), April- June, 2022, pp 25-29

PREDICTION OF SOFTWARE DEVELOPMENT EFFORTS USING ENSEMBLE APPROACH

Dr. Ripu Ranjan Sinha*
Rajani Kumari Gora**

ABSTRACT

The predominant aim of software engineering is to broaden top notch software program tasks that meet all necessities with minimum funding in budget, resources, and human resources. Estimating software development efforts are considered one of the most important tasks in software development. SDEE involves assessing the manpower, budget, and time needed to develop high quality software. SDEE accuracy enables effective planning, control, and software projects on budget and on schedule. SDEE overestimation/ underestimation is an important issue. Regular rigorous reviews are required to improve forecasts. Estimating or predicting software development efforts early in the software development life cycle helps and encourages teams to develop and deliver quality software within time and budget. Therefore, as the entire software development process relies on these predictions, the person responsible for the project manager must have the ability to accurately and reliably estimate software development effort. While software engineering experts have utilised a variety of effort estimating strategies over the last four decades, including those based on statistical and machine learning methods, no consensus has been established on which strategy performs best in all situations. Ensemble learning approaches were developed to address this problem. The ensemble model's purpose is to automatically manage each of its component model's strengths and weaknesses, resulting in the best possible decision being made overall. The main purpose of this study is to develop a model using an ensemble technique.

Keywords: *Machine Learning, Software Effort Estimation, Ensemble Techniques.*

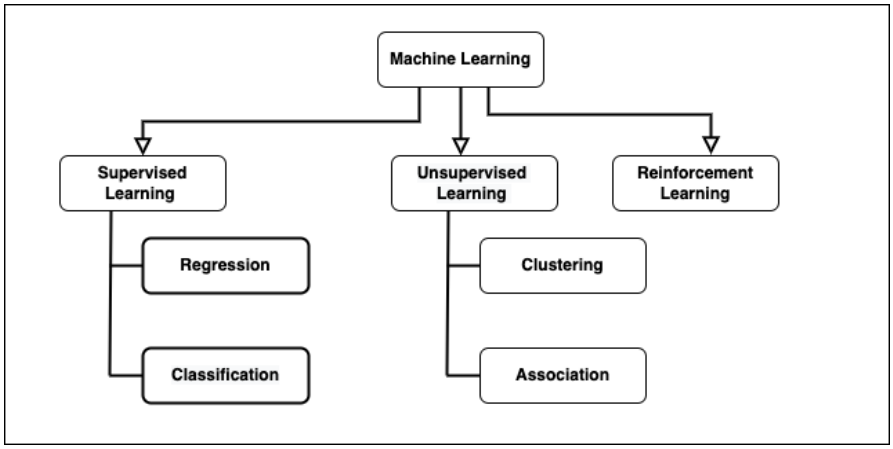
Introduction

In software project management, effort estimation is critical for predicting time and budget values in the early stages of the project, which aids in software project planning. Software estimation is considered accurate if the project estimator has a thorough understanding of the project difficulties and is able to complete the project on schedule, on budget, and to the required quality. Due to bias and subjective assessment, traditional and parametric estimation measures have proved generally wrong. If the data is appropriately pre-processed and the relevant features are chosen, data mining and machine learning methods have proven to be helpful in combating human bias and subjectivity. It discusses how to improve software estimating accuracy by employing various data pre-processing approaches, feature selection, and utilising.[1]

Machine Learning is a subset of Artificial Intelligence, which empowers the machine to naturally gain from information, further develop execution from previous encounters, and make expectations. AI contains a bunch of calculations that work on an immense measure of information. Information is taken care of to these calculations to prepare them, and based on preparing, they construct the model and play out a particular undertaking. These ML calculations help to take care of various business issues like Regression, Classification, Forecasting, Clustering, and Associations, and so forth. In view of the strategies and approach to learning, AI is separated into primarily three sorts, which are:

* Professor, S.S Jain Subodh P. G. College, Jaipur, Rajasthan, India.
** Assistant Professor, Government College, Khetri & Research Scholar, RTU, Kota, Rajasthan, India.

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning [3].



Supervised Learning (Predictive Task**)**

In supervised machine learning, the system learns from the "labelled" dataset and predicts the outcome based on the training. Some of the inputs have already been mapped to the output, as indicated by the marked data. Neural Networks, Support Vector Machines (SVM), Linear Regression, Decision Trees, and Naive Bayes are examples of regularly used algorithms. The supervised learning technique's main purpose is to translate the input variable(x) to the output variable(y) (y). Supervised machine learning is further classified into two types of problems: (i) Regression (ii) Classification

The task of approximating a mapping function (f) from input variables (X) to a continuous output variable is known as regression predictive modelling (y). A real-valued output variable, such as an integer or floating point number, is a continuous output variable. Estimating software effort is a regression problem. The task of approximating a mapping function (f) from input variables (X) to discrete output variables is known as classification predictive modelling (y). Labels or categories are terms used to describe the output variables. The mapping function determines which class or category an observation belongs to.

Unsupervised Learning (Descriptive Task**)**

In such models, training is done without supervision, which means that information is not labelled or categorised. The information is grouped by the system based on structural similarities and differences in data relationships. It looks for patterns and connections in data without labelling it. Algorithms such as k-means clustering and Association Rules are among the most often utilised.

Reinforcement Learning

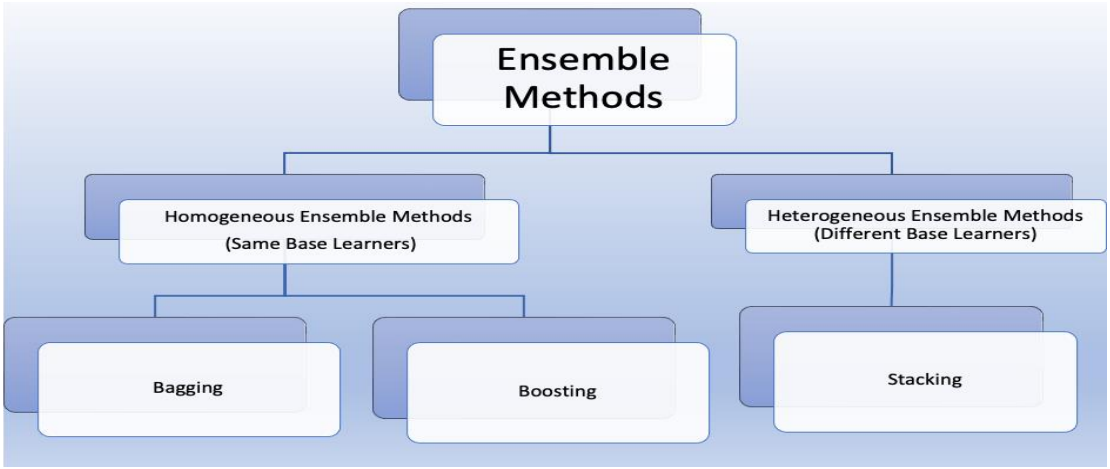
It's a type of dynamic learning in which an algorithm is taught through a combination of punishment and payoff, or reward. There is no solution in Reinforcement Learning, but the reinforcement agent takes the appropriate actions to maximise the output in the given situation. The environment provides input to the agent, which it uses to learn its behaviour.

Due to prejudice and subjectivity, traditional and parametric methods for estimating software effort are typically wrong. If the data is exposed to proper data pre-processing and feature extraction procedures, Machine Learning approaches have been found to be helpful in dealing with bias and subjectivity issues. For estimating software project timelines, machine learning (ML) approaches were used, which can adapt to changes occurring during the software project. An ensemble of numerous machine learning models would be useful to further improve estimation accuracy and stability.

Ensemble Approach

When the performance of the base models is weak, ensemble is a useful strategy. The main premise is that if methods work together as a committee with strong techniques, they can be improved and provide greater results. As a result, it is excellent for predicting software effort because each model has its own assumptions and setup parameters, resulting in an ensemble that performs incredibly well with some desirable statistical features. In practise, it is much preferable for the basis models employed

in ensemble construction to have distinct properties and behave differently. Each model makes statistically significant different predictions than the other participant models, which is referred to as diversity. In Ensemble, a new problem's solution can be derived from a set of earlier answers using either simple statistical approaches like mean, weighted mean, and inverse ranked weighted method, or more complicated machine learning-based methods like Bayesian averaging, Bagging, and boosting. Because each method in the ensemble strives to minimise and patch errors generated by other techniques, the methods in ensembles can boost each other, lowering estimate errors. The ensemble model's pooled findings should have superior overall accuracy than any single member, according to the principle.



A homogeneous ensemble is made up of members who all use the same sort of basic learning method. The structure can make a difference in the ensemble members in this scenario [1]. Heterogeneous ensemble consists of members having different base learning algorithms. Bagging and boosting are homogeneous ensemble methods that aggregate the outcomes of the same learners, whereas stacking uses an algorithm to combine the results of several learners. Stacking is a powerful heterogeneous ensemble of machine learning techniques used to improve accuracy. The performance of the stacking ensemble methods will be determined by the learners' selection and adjustment of their hyperparameters.

Literature Review

Idri et al. [4] performed an SLR based on 24 EEE studies published between 2000 and 2016 and the review was made with six viewpoints: ensemble estimation accuracy, accuracy comparison of EEE techniques with a single model, methodologies used to construct ensemble methods, the single model used to construct ensembles, rules used to combine single estimates, and accuracy comparison among EEE techniques. The review concluded that EEE techniques may be separated into two types homogeneous and heterogeneous and that the ML single model is frequently employed in constructing EEE. The result of the study shows that EEE usually yields acceptable estimation accuracy and is more accurate than single models.

Elish [5] evaluated the extent to which the voting ensemble model, with median combination rule, offers reliable and improved estimation accuracy over five individual models: MLP, RBF, RT, K-nearest neighbour (KNN), and SVR in estimating software development effort. In three out of the five datasets that were used in that study, the ensemble model outperformed the individual models. In the other two datasets, the ensemble model achieved the second-best performance. The results confirm that individual models are not reliable as their performance is inconsistence and unstable across different datasets. However, the ensemble model provides more reliable performance than individual models.

W. L. Jiang et al [6], developed a self-adaptive stacking ensemble model (SSEM) by integrating base algorithms which have low complexity and high diversity. This model takes care of the selection of “good and different” base-classifiers and assigns appropriate parameters to every base-classifier. Minimum mean-square error as the fitness function and genetic algorithm (GA) is used to determine the optimal parameters of the integrated base-classifiers. The model was validated on 9 different datasets and the result show that it performed better in comparison to other classification methods.

Elish et al [7], evaluated and compared different homogeneous and heterogeneous ensembles models for software development effort estimation. The results verify that individual models are not reliable as their performance is inconsistent and unstable across various datasets. Although none was the best, as every dataset is having its own best models.

Sikora et al [8], proposed a modified stacking ensemble algorithm using Genetic Algorithm (GA). J48, Naive Bayes, NN, IBK, OneR algorithms were used as base models and Genetic algorithms was used as meta model. 10 different datasets of the UCI Data Repository Were used in experiment. The model shows improvement in the performance over the individual learning algorithms as well as over the standard stacking algorithm.

Research Background

The information we used came from the Promise data source. There are 19 features in the China dataset, with one dependent and eighteen independent variables. However, some of the independent variables are omitted because they are unimportant in predicting effort, resulting in a much simpler and efficient model. Data dimensionality can be reduced using a variety of ways. To minimise the number of independent variables, we employed the feature sub selection strategy given in the WEKA programme [11]. The 19 variables were reduced to 10 after adopting Correlation Based Feature Subselection (CFS) (one dependent and nine independent variables). The correlation-based feature selection technique (CFS) is used to pick the best predictors from the datasets' independent variables. Through all potential combinations of variables, the best independent variable combinations were found. CFS assesses the best of a group of variables by taking into account each feature's individual predictive power as well as the degree of redundancy between them. Effort is the dependent variable. Software development effort is defined as the work carried out by the software supplier from specification until delivery measured in terms of hours .

The independent variables are Output, Enquiry, Interface, Added, PDR_AFP, PDR_UFP, NPDR_AFP, NPDU_UFP and Resource. All the independent variables correspond to function point method.

Performance Measures

We have used the following evaluation criterion to evaluate the estimate capability.

- Mean Magnitude of relative error (MMRE) (or mean absolute relative error)

$$\mathrm{MMRE} = \frac{1}{n}\sum_{i=1}^{n} \left|\mathrm{Pi} - \mathrm{Ai}\right| \frac{1}{Ai}$$

Where Pi is the predicted value for datapoint i; Ai is the actual value for datapoint i; n is the total number of datapoints .

- Root Mean Squared Error (RMSE)

The root mean squared error is defined as:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\mathrm{Pi} - \mathrm{Ai})^{2}}$$

Where Pi is the predicted value for datapoint i; Ai is the actual value for datapoint i; n is the total number of datapoints If Pi =Ai , i = 1,2, ..., n; then E=0 (ideal case)

- **Mean Absolute Error**

The mean absolute error measures of how far the estimates are from actual values. It could be applied to any two pairs of numbers, where one set is "actual" and the other is an estimate, prediction.

- **Correlation Coefficient**

The strength of a relationship between two variables is measured by correlation. The correlation coefficient indicates the strength of the link. The stronger the association, the higher the correlation coefficient value.

- **Validation Measures**

There are three types of validation techniques: hold-out, leave-one-out, and K-cross. Because our dataset is huge (499 data points), we utilise the hold out approach, which divides the dataset into two parts: training and validation.

Result Analysis

In this study, we use machine learning approaches such as Multiple Linear Regression to predict effort. The effort estimation model was predicted using the China Dataset [10]. The accuracy of the effort estimation model was estimated using a holdout technique of cross validation. In a 7:3 ratio, the dataset was partitioned into two parts: training and validation. As a result, 70% of the money was spent on training the model and 30% on evaluating its correctness. Among the models, the one with the lowest MMRE, RMSE, MAE, and greater correlation coefficient is regarded the best.

Performance Measures	Linear Regression
Mean Magnitude Relative Error (MMRE) %	17.97
Root mean squared error (RMSE) %	4008.11
Correlation coefficient	0.79
Mean absolute error (MAE) %	1981.48

Conclusion

For this study, regression was utilised to predict effort. Using data from the Promise data repository, we were able to achieve findings. With the CFS technique, we were able to reduce the dataset from 19 to 10 characteristics. The findings suggest that the machine learning method used in this work is capable of producing a model that accurately predicts maintenance effort.

References

1. Mahmoud O. Elish, Tarek Helmy, Muhammad Imtiaz Hussain, "Empirical Study of Homogeneous and Heterogeneous Ensemble Models for Software Development Effort Estimation", *Mathematical Problems in Engineering*, vol. 2013, Article ID 312067, 21 pages, 2013. https://doi.org/10.1155/2013/312067
2. J. Moudrík and R. Neruda, "Evolving Non-Linear Stacking Ensembles for Prediction of Go Player Attributes," 2015 IEEE Symposium Series on Computational Intelligence, 2015, pp. 1673-1680, doi: 10.1109/SSCI.2015.235.
3. Sinha R R, Gora R K (AICTC-2019), "Advances in Information Communication Technology and Computing", Lecture Notes in Networks and Systems 135, Springer Nature, https://doi.org/10.1007/978-981-15-5421-6_8
4. Idri A, Hosni M, Abran A (2016), "Systematic literature review of ensemble effort estimation", The Journal of System and Software, https://doi.org/10.1016/j.jss.2016.05.016
5. M. Elish (2013), "Assessment of voting ensemble for estimating software development effort," in Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM '13), pp. 322–327.
6. Weili Jiang, Zhenhua Chen, Yan Xiang, Dangguo Shao, Lei Ma, Junpeng Zhang (2019), "SSEM: a novel self-adaptive stacking ensemble model for classification", IEEE, 10.1109/ACCESS.2019.2933262
7. M. Elish, T. Helmy, M. I. Hussain (2013), "Empirical Study of Homogeneous and Heterogeneous Ensemble Models for Software Development Effort Estimation" Mathematical problems in engineering, Hindawi Publishing Corporation. http://dx.doi.org/10.1155/2013/312067
8. Sikora, Riyaz and Al-laymoun, O'la Hmoud (2014) "A Modified Stacking Ensemble Machine Learning Algorithm Using Genetic Algorithms," Journal of International Technology and Information Management: Vol. 23: Iss. 1, Article1 https://scholarworks.lib.csusb.edu/jitim/vol23/iss1/1.
9. Malhotra, R., & Jain, A. (2011). Software effort prediction using statistical and machine learning methods. *International Journal of Advanced Computer Science and Applications*, *2*(1).
10. Promise. Available: http://promisedata.org/repository/.
11. Weka. Available:http://www.cs.waikato.ac.nz/ml/weka/.

❑❍❑