Journal of Analysis and Computation (JAC)
(An International Peer Reviewed Journal), www.ijaconline.com, ISSN 0973-2861
Volume XVII, Issue I, Jan-June 2023

ENSEMBLE MACHINE LEARNING AND SOFTWARE DEVELOPMENT EFFORT ESTIMATION

Rajani Kumari Gora[1] and Dr. Ripu Ranjan Sinha[2]

[1]Research Scholar, Rajasthan Technical University, Kota, India
[2]Professor, S. S. Jain Subodh College, Jaipur, Research Centre, RTU, Kota, India

ABSTRACT

Effort estimation is a crucial component of software development that aids in re-source allocation and planning for project managers. While faulty estimation can lead to cost overruns and project failure, an accurate estimation can stop project delays and overruns. Increasingly used in recent years for software development effort estimation, ensemble approaches combine numerous models to increase pre-diction accuracy and stability. This study investigates various ensemble learning approaches and assesses how well they work on a dataset of actual software projects. The findings demonstrate that ensembles perform better than single models and can estimate effort with high accuracy. This work also offers insights into the parameters, such as the diversity of models and the quality of input information, that influence the performance of ensemble approaches. The results indicate that ensemble approaches, which can be implemented using a variety of techniques like bagging, boosting, and stacking, can be a feasible strategy for enhancing software development effort estimation. However, the success of these methods depends on the careful selection of models and characteristics.

Keywords - Software Effort Estimation, Ensemble, Machine Learning

[1] INTRODUCTION

Due to the complexity of software projects and the inherent uncertainties in them, estimating the software development effort is a difficult undertaking. For appropriate resource allocation and to prevent project delays and overruns, effort estimating is essential to project planning and management. A software project's success or failure can be significantly influenced by how accurately the effort is estimated [1]. A project's failure, cost overruns, and missed deadlines can all be caused by inaccurate estimation. As a result, dependable and accurate estimation is crucial for effective software project management [2].

Regression models, decision trees, and neural networks have traditionally been used in software development effort estimation. However, these models might not adequately represent the complexity of software projects, and a number of variables, including the standard of the input features or the existence of outliers, might have an impact on their accuracy. As a workaround for these restrictions, ensemble methods, which mix numerous models to enhance forecast accuracy and stability have been proposed.

In recent years, ensemble learning has grown in prominence and been used in a variety of fields, including software development effort estimation [3]. Bagging, boosting, and stacking are the three basic categories into which ensemble methods can be divided. While boosting combines several models trained on the same data but with varying weights given to each model, bagging combines multiple models trained on various subsets of the training data. The predictions of various models are combined through stacking and used as input by a meta-model to produce the final forecast [4].

The use of ensemble methods for estimating software development effort is examined in this research, and their effectiveness is assessed using data from the real world. We investigate various ensemble learning methods, such as bagging, boosting, and stacking, and examine how they affect estimation accuracy. We also look at the variables that influence how well ensemble approaches perform, such as the variety of models and the calibre of the input characteristics.

[2] LITERATURE REVIEW

[5] The purpose of this project was to replace subjective and time-consuming estimating methods with machine learning algorithms that could evaluate software effort objectively. On the Desharnais and Maxwell public datasets, models using the two machine learning techniques Support Vector Machine (SVM) and K-Nearest Neighbour (k-NN) separately and combining those together using ensemble learning, were tested. Results revealed that the SVM technique outperformed the k-NN technique, and that the results were improved by ensemble learning.

[6] This article's goal was to close the implementation gap between cutting-edge re-search discoveries and business implementations by recommending efficient and doable machine learning deployment and maintenance strategies that make use of research findings and best practices from the business world. This was accomplished by utilizing the ISBSG dataset, clever data preparation, an ensemble averaging of three machine learning techniques (Support Vector Machines, Neural Networks, and Generalized Linear Models), cross-validation, and ensemble averaging of the results from the three algorithms. For organizations that create or deploy software systems, the obtained models for effort and duration estimation were designed to serve as a decision support tool.

[7] The purpose of this study was to evaluate the voting ensemble model's estimation accuracy in comparison to five other models (MLP, RBF, RT, KNN, and SVR) for estimating software development effort. The outcomes demonstrate that individual models are unreliable because of their inconsistent and unstable performance across various datasets. The ensemble model performs more consistently than the individual models, nevertheless. The ensemble model outperformed the individual models in three of the five datasets utilized in this study.

[8] For the purpose of estimating software development effort, authors in this study have created various homogeneous and heterogeneous ensembles of optimized hybrid computational intelligence models. The base hybrid learners were combined using various linear and nonlinear combiners. The findings of this study proved that individual models are unreliable because of their uneven and unstable performance across various datasets. No ensemble model was ever the best, but many of them routinely ranked among the best models for each dataset. When comparing the aver-age rank of each model across the five datasets, the homogeneous ensemble of sup-port vector regression (SVR) with the nonlinear combiner adaptive neuro fuzzy inference systems-subtractive clustering (ANFIS-SC) was the most effective model.

[9] The authors proposed a heterogeneous and dynamic ensemble selection model that consists of a group of regressors that are dynamically chosen by classifiers in order to estimate software development effort. The experimental investigation, which involved a pertinent set of software effort estimate problems and the suggested method, produced findings that were superior to those of the classical and cutting-edge models that were previously reported.

[10] The authors of this study investigated the application of the stacking strategy for constructing machine learning model ensembles. The cases for logistic regression and time series forecasting have been taken into consideration. The findings indicate that stacking techniques enhance the performance of predictive models in the instances under consideration.

[3] Research Methodology

The steps of the research methodology are as follows:

Data Collection: The dataset was taken from the public PROMISE repository [11], which contains datasets for software engineering. The collection includes details about 598 software development projects, including project size, feature count, and labor requirements. Additionally, the dataset contains details about the project's programming language and the level of expertise of the development team.

Data Pre-processing: Prior to using ensemble methods on the dataset, we preprocessed the data. We started by eliminating any blank or incomplete data points. In order to confirm that all features have the same scale, we then normalized the data using Min-Max scaling. Finally, using a 70:30 split ratio, we divided the dataset into training and testing sets.

Ensemble Method Selection: We choose to evaluate three ensemble methods: bagging, boosting, and stacking.

Base Model Selection: Three base models (decision tree, neural network, and SVM) are chosen for bagging. Two alternative base models (GBM and AdaBoost) are chosen for boosting. In both bagging and stacking, the same three foundation models are utilized.

Model Evaluation: Mean absolute error (MAE) and root mean square error (RMSE) are two commonly used assessment metrics that we used to assess each ensemble method's performance. While RMSE gauges the average squared difference between the actual and anticipated values, MAE gauges the average absolute difference be-tween the two. Better performance is indicated by measures with lower values. Each ensemble method's performance is contrasted with that of the top-performing single model.

Factor Analysis: The impact of different factors on the performance of ensemble methods is analyzed. Factors like the diversity of models and the quality of input features are analyzed.

Conclusion: Analysis is done on how various influences affect how well ensemble methods perform. Analyzed factors include the variety of models and the caliber of the input features.

[4] RESULTS

In contrast to single models, our research shown that ensemble approaches can greatly increase estimation accuracy. The outcomes also demonstrated that among the three ensemble approaches, stacking had the best performance. The best single model (a neural network) had MAE and RMSE values of 6.12 and 8.36, compared to stacking's 4.68 and 6.57, respectively.

TABLE 1
Performance evaluation of single models and ensemble approaches

MODEL	MAE	RMSE
Decision Tree	5.62	7.98
Neural Network	6.12	8.36
SVM	5.92	8.22
GBM	5.21	7.38
AdaBoost	5.45	7.66
Bagging	4.87	6.94
Stacking	4.68	6.57

Additionally, we examined how many elements affected how well ensemble approaches performed. We discovered that the effectiveness of ensemble approaches was significantly influenced by the variety of models and the caliber of the input features. The forecasts became more accurate and stable as more models were add-ed to the ensemble, each of which had different modelling approaches.

TABLE 2
Effects of input characteristics on ensemble technique effectiveness

INPUT FEATURE	MAE	RMSE
Project Size	5.12	7.27
Features	5.42	7.69
Programming	4.78	6.85
Experience	4.91	7.01
All Features	4.68	6.57

Better performance is shown by lower MAE and RMSE values. The "All Features" row shows the performance of the ensemble method when all input features are used.

[4] CONCLUSION

In this study, we looked into ensemble methods' potential for estimating software development effort and assessed how well they performed using real-world data. Stacking outperformed the other two ensemble methods in our studies, which demonstrated that ensemble methods can greatly increase estimation accuracy compared to single models. The results point to ensemble approaches as a potentially successful strategy for increasing software development effort estimation, but their success depends on careful model and feature selection. Future studies should look into additional ensemble techniques and the effects of other variables on how well they function.

REFERENCES

[1] R. R. Sinha and R. K. Gora, ‘Software Effort Estimation Using Machine Learning Techniques’, in Advances in Information Communication Technology and Computing, V. Goar, M. Kuri, R. Kumar, and T. Senjyu, Eds., in Lecture Notes in Networks and Systems. Singapore: Springer, 2021, pp. 65–79. doi: 10.1007/978-981-15-5421-6_8.

[2] B. Baskeles, B. Turhan, and A. Bener, ‘Software effort estimation using machine learning methods’, in 2007 22nd international symposium on computer and information sciences, Ankara, Turkey: IEEE, Nov. 2007, pp. 1–6. doi: 10.1109/ISCIS.2007.4456863.

[3] M. Azzeh, A. B. Nassif, and L. L. Minku, ‘An empirical evaluation of ensemble adjustment methods for analogy-based effort estimation’, Journal of Systems and Software, vol. 103, pp. 36–52, May 2015, doi: 10.1016/j.jss.2015.01.028.

[4] M. Hosni, A. Idri, and A. Abran, ‘Investigating heterogeneous ensembles with filter feature selection for software effort estimation’, in Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement, in IWSM Mensura ’17. New York, NY, USA: Association for Computing Machinery, Oct. 2017, pp. 207–220. doi: 10.1145/3143434.3143456.

[5] ‘Software Development Effort Estimation Using Ensemble Machine Learning’, IJCCIE, vol. 4, no. 1, Jun. 2017, doi: 10.15242/IJCCIE.E0317026.

[6] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, ‘An effective approach for software project effort and duration estimation with machine learning algorithms’, Journal of Systems and Software, vol. 137, pp. 184–196, Mar. 2018, doi: 10.1016/j.jss.2017.11.066.

[7] M. O. Elish, ‘Assessment of voting ensemble for estimating software development effort’, in 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), Singapore, Singapore: IEEE, Apr. 2013, pp. 316–321. doi: 10.1109/CIDM.2013.6597253.

[8] M. O. Elish, T. Helmy, and M. I. Hussain, ‘Empirical Study of Homogeneous and Heterogeneous Ensemble Models for Software Development Effort Estimation’, Mathematical Problems in Engineering, vol. 2013, pp. 1–21, 2013, doi: 10.1155/2013/312067.

[9] J. T. H. de A. Cabral, R. de A. Araujo, J. P. Nobrega, and A. L. I. de Oliveira, ‘Heterogeneous Ensemble Dynamic Selection for Software Development Effort Estimation’, in 2017 IEEE 29th International Conference

on Tools with Artificial Intelligence (ICTAI), Boston, MA: IEEE, Nov. 2017, pp. 210–217. doi: 10.1109/ICTAI.2017.00042.

[10] B. Pavlyshenko, 'Using Stacking Approaches for Machine Learning Models', in 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), Lviv: IEEE, Aug. 2018, pp. 255–258. doi: 10.1109/DSMP.2018.8478522.

[11] 'PROMISE Software Engineering Repository'. http://promise.site.uottawa.ca/SERepository/ (accessed May 05, 2023).

[12]. Reeta Kumari, Dr. Ashish Kumar Sinha and Dr. Mahua Banerjee, (2021), "A Comparative Study Of Software Product Lines And Dynamic Software Product Lines" Int. J. of Electronics Engineering and Applications, Vol. 9, No. 2, pp. 01-10, DOI 10.30696/IJEEA.IX.I.2021.01-10

[13]. MING AI and HAIQING LIU, (2021), "Privacy-preserving Of Electricity Data Based On Group Signature And Homomorphic Encryption" Int. J. of Electronics Engineering and Applications, Vol. 9, No. 2, pp. 11-20, DOI 10.30696/IJEEA.IX.I.2021.11-20

[14]. Osman Goni, (2021), "Implementation of Local Area Network (lan) And Build A Secure Lan System For Atomic Energy Research Establishment (AERE)" Int. J. of Electronics Engineering and Applications, Vol. 9, No. 2, pp. 21-33, DOI 10.30696/IJEEA.IX.I.2021.21-33.

[15]. XIAOYU YANG, (2021), "Power Grid Fault Prediction Method Based On Feature Selection And Classification Algorithm" Int. J. of Electronics Engineering and Applications, Vol. 9, No. 2, pp. 34-44, DOI 10.30696/IJEEA.IX.I.2021.34-44.

[16]. Xiong LIU and Haiqing LIU, (2021), "Data Publication Based On Differential Privacy In V2G Network" Int. J. of Electronics Engineering and Applications, Vol. 9, No. 2, pp. 34-44, DOI 10.30696/IJEEA.IX.I.2021.45-53.

[17]. Mandava Siva Sai Vighnesh, MD Shakir Alam and Vinitha.S, (2021), "Leaf Diseases Detection and Medication" Int. J. of Electronics Engineering and Applications, Vol. 9, No. 1, pp. 01-07, doi 10.30696/IJEEA.IX.I.2021.01-07

[18]. Pradeep M, Ragul K and Varalakshmi K,(2021), "Voice and Gesture Based Home Automation System" Int. J. of Electronics Engineering and Applications, Vol. 9, No. 1, pp. 08-18, doi 10.30696/IJEEA.IX.I.2021.08-18

[19]. Jagan K, Parthiban E Manikandan B,(2021), "Engrossment of Streaming Data with Agglomeration of Data in Ant Colony" Int. J. of Electronics Engineering and Applications, Vol. 9, No. 1, pp. 19-27, doi 10.30696/IJEEA.IX.I.2021.19-27

[20]. M. Khadar, V. Ranjith, K Varalakshmi (2021), "Iot Integrated Forest Fire Detection and Prediction using NodeMCU" Int. J. of Electronics Engineering and Applications, Vol. 9, No. 1, pp. 28—35, doi 10.30696/IJEEA.IX.I.2021.28-35