International Journal of Engineering Research & Management Technology
ISSN: 2348-4039
November- December 2023 Volume 10, Issue-6
Email: editor@ijermt.org www.ijermt.org

"SOFTWARE DEVELOPMENT EFFORT ESTIMATING USING MACHINE LEARNING"

Vasudeva Rao P V[1]

[1]Research Scholar, Department of Computer Science and Engineering, Kalinga University Raipur, Chhattisgarh, India

Dr. Dev Ras Pandey[2]

[2]Assistant Professor, Department of Computer Science and Engineering, Kalinga University Raipur, Chhattisgarh, India

Abstract

Estimating the amount of work that goes into making software is an important part of project management because it has a direct effect on planning, scheduling, and allocating resources. Because software projects are so complicated and changeable, traditional ways of estimating, like expert opinion and algorithmic models like COCOMO, don't always work well. Using old project data to find patterns and correlations, machine learning (ML) has become a hopeful way to improve the accuracy of estimates. This essay looks at several machine learning methods, such as regression models, decision trees, support vector machines, and neural networks that can be used to guess how much work it will take to make software. We talk about choosing the right features, preparing the data, and using model evaluation metrics to make predictions more accurate. We show that ML-based models can do better than traditional estimation methods by using real-world datasets to make more accurate and flexible guesses about how much work needs to be done. The study shows that machine learning has the ability to cut down on estimation mistakes, make project planning more efficient, and make software development more efficient overall. To get the most out of ML's benefits in effort estimation, though, problems like bad data, models that are hard to understand, and variations that are specific to the topic must be fixed.

Keyword: Estimating software work, machine learning, project management, regression models, neural networks, data-driven estimation, and predictive analytics are some of the topics that this course covers.

1. INTRODUCTION

Estimating the amount of work that would be required to complete a software project is something that software teams and companies have struggled with for a long time. Software project success and risk mitigation depend

on precise software work estimating. Effort estimate is the technique of predicting the time and resources needed to create a software process or product. Accurately forecasting software expenses is essential for good estimating, planning, controlling, and managing the project. It is critical to have accurate time and cost estimates in order to plan for software development and allocate resources efficiently. Because the time and money constraints needed to complete the project are calculated during project planning, the success or failure of the project is determined at this point. With the advent of the computer industry in the 1940s, the concept of software effort estimate began to gain traction. This field of study is far from finished. In research on software effort estimates, many different approaches of estimating are classified into three main categories: algorithmic, non-algorithmic, and machine learning.

When estimating the cost of a software project, algorithmic methods turn to mathematical and statistical principles. Some examples of estimation methodologies include COCOMO-II, SEER-SEM, True Planning, and PutnamSoftware Life Cycle Management (SLIM). The size of the program under evaluation, often measured in terms of function points, source lines of code, or use case points, is the primary input to these models. Without algorithms, models depend on human judgments and interpretations of data. Using data from previous research, these models draw conclusions. Methods that do not rely on algorithms include expert opinion, planning poker, wide-band Delphi, and the work breakdown structure (WBS). One substitute for creating models algorithmically is machine learning.

Machine learning estimation methodologies include things like artificial neural networks (ANN), decision trees (DT), fuzzy models, Bayesian networks, genetic algorithms, case-based reasoning (CBR), and support vector regression (SVR). In order to quantify the estimated effort of software development, many datasets have been suggested. These datasets have been suggested for some time. A lot has changed recently in terms of effort participation in software development. Many businesses have adopted a hybrid development strategy in the post-COVID age, according to one point of view. Developers in hybrid mode are not need to physically visit the office on a daily basis. Home is their office. It is only necessary to attend the office if absolutely necessary.

2. LITERATURE REVIEW

Rahman, Mizanur& Roy, Partha& Ali, Mohammad &Gonc¸alves, Teresa &Sarwar, Hasan (2023) This paper discusses the importance of accurate effort estimation in software development projects. It highlights the use of machine learning techniques and algorithms, such as decision trees, k-nearest neighbor regression, and support vector regression, to assess predictions more effectively. These techniques have gained interest due to

issues with parametric and conventional estimate methods and advertising campaigns. The paper uses a dataset from Edusoft Consulted LTD to measure the standard procedure's performance using metrics like mean squared error, R-squared error, and mean absolute error. Comparative experimental evidence suggests that decision trees are the best strategy for assessing effort.

Un Nisa, Mehar&Saqlain, Muhammad &Naeem, Abid Amin &Awais, Muhammad &Stević, Željko. (2023) This study focuses on improving software effort estimate using machine learning algorithms and datasets. The study uses publicly available datasets such as ISBSG, NASA93, COCOMO, Maxwell, and Desharnais. Data is separated into train and test sets, and missing value management and categorical feature conversion are performed. Four machine learning regression techniques are tested: Decision Tree, Gradient Boosting, Linear Regression, and Random Forest. The dimensionality is reduced, and appropriate subsets of characteristics are chosen using correlation-based feature selection. The precision of predictions is assessed using R2 and RMSE metrics. The results show that Random Forest and linear regression models outperform alternative techniques for this effort estimating job. The NASA93, COCOMO, Maxwell, and Desharnais datasets have the highest R2 scores, while the Desharnais dataset has the lowest RMSE. The study suggests that improving machine learning models for software effort estimate using correlation-based feature selection can enhance accuracy. The findings provide a solid foundation for other studies and can be used by software project planners to create smart effort prediction systems that are data-driven.

Sousa, André &Veloso, Daniel &Gonçalves, Henrique &Faria, João& Moreira, João&Graça, Ricardo & Gomes, Duarte & Castro,Rui&Henriques, Pedro. (2023) This study aims to determine the best machine learning algorithms for software estimating in project management, focusing on task-specific estimates. Data was collected from three different project management software and eight machine learning methods were used to train regression models. The models were validated using k-fold cross-validation and assessed using various metrics. Ensemble algorithms such as XGBoost, Random Forest, and Extra Trees Regressor performed better than non-ensemble methods on all three datasets. The feature significance and estimate accuracy varied among datasets, with MMRE values ranging from 0.11 to 9.45 for target variables and datasets combined. However, with MMRE = 0.23, effort estimates aggregated to the project level demonstrated high accuracy even in the worst-performing dataset. Machine learning techniques, particularly ensemble ones, seem to be a good solution for software project task time and effort estimation. Dataset and project specifics may impact estimate quality and relevant feature identification, but project-level aggregated predictions may still show a respectable degree of accuracy due to error compensation.

Banimustafa, Ahmed. (2018) Software development relies heavily on estimates, which can impact project success by changing costs and effort. Algorithms like COCOMO, Function Point Analysis, and Use-Case-Points can lead to overbudgeting and behind schedules. To improve estimate, data mining using historical data is suggested. Three machine learning techniques—Naïve Bayes, Logistic Regression, and Random Forests—are applied to preprocessed COCOMO NASA benchmark data from 93 projects. The models were assessed using Classification Accuracy, Precision, Recall, and AUC, and tested using five-fold cross-validation.

Varshini, Priya&Kumari K, Anitha&Janani, D & .S, Soundariya (2021) Machine learning and deep learning are crucial for artificial intelligence, enabling the creation of problem-solving intelligent systems. Software effort estimation is used to estimate labor hours needed for a project, which can be challenging due to unknowns. Various algorithms, including deepnet, neuralnet, support vector machine, and random forest, are used to predict effort. The study compares these algorithms, finding random forest as the most effective due to its resilience and ability to handle large datasets. Evaluation metrics such as Mean Absolute Error, Root Mean Squared Error, Mean Square Error, and R-Squared are discussed.

3. STATEMENT OF AIM

Software development effort estimation has seen a rise in the application of ensemble learning within the last 20 years. Ensembles have been the subject of several experiments. Common ensemble procedures like averaging, voting, bagging, etc., have been used by them. Few have tried out new ensemble techniques like as layered generalization, the AdaBoost algorithm, or gradient tree boosting. Similarly, when it comes to predicting software development effort using ensemble models, very few research has explored with parameter tweaking and feature selection strategies. The probability of obtaining better outcomes is increased by incorporating these strategies. We are motivated to investigate and evaluate several ensemble approaches in order to create an effective ensemble model for software development effort prediction by the encouraging outcomes of the aforementioned papers and publications.

4. NEED OF THE STUDY

LIMITATION

- The scarcity of publicly accessible data meant that we were unable to compare our findings to those of other research; this was one of the constraints we encountered. Possible causes of this data scarcity include

concerns about privacy related to software project information, a lack of documentation of metrics used in software development, and other similar issues.

- In a similar vein, we were unable to compare our study's findings to those of other researchers as their tools did not work with the platform or console that we used to build our model.

- The datasets' sizes and the features' data types are noticeably different from one another. One potential drawback is that the generated model's performance could be affected by these changes, and thus performance might differ among datasets.

DELIMITATION

According to Denscombe (2013), researchers use delimitations to clearly define the scope of their study. Small and medium-sized software development companies were the only ones included in the research. A second constraint was that the participants were only from software development teams that were involved in the estimating phase of the research. Also, I only included software development organizations in the South Texas region in this analysis.

5. OBJECTIVES

1. Identification of various Machine Learning Techniques, Datasets which were used earlier in estimating Software Development Effort.

2. Determining whether the developed Ensemble model performs better than the existing models (both individual Machine Learning techniques and Ensemble Models).

3. Comparison of outputs/predictions made by the developed Ensemble model while using parameter tuning and feature selection.

4. Determining efficiencies by comparing outputs/predictions made by the Ensemble model with multiple datasets.

6. RESEARCH METHODOLOGY

The historical dataset's features and quality determine the effectiveness of an effort-estimating model. Similar to before, ABSDEE compares projects' qualities to determine which are comparable. To locate the most

comparable previously completed projects, it will be more effective to utilize just the subset of the dataset that will be likely to include more of them rather than utilizing the full dataset for comparison. Next, we will conduct a controlled experiment to build an ensemble model and evaluate its performance against existing models. This evaluation would reveal if the generated model outperformed the competitors.

Data Collection

We will source the selected datasets from the "Predictor Models in Software Engineering (PROMISE)" database, which provides freely accessible datasets for software engineering research and study. Furthermore, using Google's dataset search and Kaggle, we will obtain the necessary datasets for the experiment.

Datasets

This thesis will make use of the following datasets:

COCOMO81

ALBERCHT

MAXWELL

DESHARNAIS

7. HYPOTHESIS

H0: No difference, in the performance of ensemble model build in this study (by combining multiple machines leaning techniques) while comparing with the machine learning techniques, ensemble techniques identified in RQ1.

H1: There is a difference in the performance of ensemble model build in this study (by combining multiple machines leaning techniques) while comparing with the machine learning techniques, ensemble techniques identified in RQ1.

H0: No difference, in the performance of ensemble model build in this study (by combiningmultiple machines leaning techniques) when implemented with parameter tuning.

H1: There is a difference in the performance of ensemble model build in this study (bycombining multiple

machines leaning techniques) when implemented with parameter tuning

8. STATEMENT ON EXPECTED ANALYSIS AND INTERPRETATION

Converting categorical data into ordinal and standardizing continues values is considered as an important step in the experiment, as we are dealing with datasets that contain multiple datatypes.we can observe that COCOMO81 dataset has numeric and continuous type of data. we handle datasets having only numeric and continues datatypes by directly standardizing. The data is standardized after dividing the dataset into two parts, 'target' storing the target variable, 'source' storing the rest of the variables and before splitting the data for training and testing. Data before and after standardization can be clearly seen in the below figure:

Figure: 1. Before Standardization of COCOMO81 dataset

	rely	data	cplx	time	stor	virt	turn	acap	aexp	pcap	vexp	lexp	modp	tool	sced	loc	actual
0	0.88	1.16	0.70	1.00	1.06	1.15	1.07	1.19	1.13	1.17	1.10	1.00	1.24	1.10	1.04	113.0	2040.0
1	0.88	1.16	0.85	1.00	1.06	1.00	1.07	1.00	0.91	1.00	0.90	0.95	1.10	1.00	1.00	293.0	1600.0
2	1.00	1.16	0.85	1.00	1.00	0.87	0.94	0.86	0.82	0.86	0.90	0.95	0.91	0.91	1.00	132.0	243.0
3	0.75	1.16	0.70	1.00	1.00	0.87	1.00	1.19	0.91	1.42	1.00	0.95	1.24	1.00	1.04	60.0	240.0
4	0.88	0.94	1.00	1.00	1.00	0.87	1.00	1.00	1.00	0.86	0.90	0.95	1.24	1.00	1.00	16.0	33.0

cplx	time	stor	virt	turn	acap	aexp	pcap	vexp	lexp	modp	tool	sced	loc	actual
-1.947900	-0.709752	-0.470864	1.183526	1.223167	1.894626	1.533724	1.407765	1.023004	-0.027700	1.815920	0.976066	-0.118544	0.214099	0.750763
-1.201442	-0.709752	-0.470864	-0.070322	1.223167	0.630486	-0.326067	0.378607	-1.136100	-0.997183	0.738100	-0.199692	-0.651992	1.290868	0.507275
-1.201442	-0.709752	-0.807960	-1.156989	-0.395208	-0.300986	-1.086891	-0.468935	-1.136100	-0.997183	-0.724657	-1.257874	-0.651992	0.327758	-0.243666
-1.947900	-0.709752	-0.807960	-1.156989	0.351735	1.894626	-0.326067	2.921233	-0.056548	-0.997183	1.815920	-0.199692	-0.118544	-0.102950	-0.245326
-0.454984	-0.709752	-0.807960	-1.156989	0.351735	0.630486	0.434756	-0.468935	-1.136100	-0.997183	1.815920	-0.199692	-0.651992	-0.366161	-0.359876

Figure:2. After Standardization of COCOMO81 dataset

9. EXPECTED OUTCOMES

Project Cost Assessment for Software work will be a regression-type issue that estimates the overall work

needed to construct a software project. In software estimation, the SDEE models will serve as a decision-support system for the project manager. Flexibility and robustness in functioning effectively with various data should be the primary concerns of any estimating model builder. The data type or dataset we will be working with dictates the use, selection, and performance of machine learning algorithms. We will adapt machine learning algorithms based on the nature of the issue at hand, since no one method can handle all of them [86]. Choosing the right machine learning approach is crucial for efficient issue resolution. This implies that the methods employed to address issues, such as SDE estimations, may vary across projects and organizations. To aid project managers in making more precise SDE estimates, a model that incorporates the best features of several machine learning approaches might be useful. The comprehension of the dataset will be critical, just as important as the ideation of the experimental design and the estimate model. One intriguing way to increase the estimate accuracy of SDEE models will be to identify key material in datasets. This will improve dataset quality.

10. REFERENCES

1. Rahman, Mizanur& Roy, Partha& Ali, Mohammad &Gonc¸alves, Teresa &Sarwar, Hasan. (2023). Software Effort Estimation using Machine Learning Technique. International Journal of Advanced Computer Science and Applications. 14. 10.14569/IJACSA.2023.0140491.

2. Un Nisa, Mehar&Saqlain, Muhammad &Naeem, Abid Amin &Awais, Muhammad &Stević, Željko. (2023). Analysis of Software Effort Estimation by Machine Learning Techniques. Ingénierie des systèmes d information. 28. 1445-1457. 10.18280/isi.280602.

3. Sousa, André &Veloso, Daniel &Gonçalves, Henrique &Faria, João& Moreira, João&Graça, Ricardo & Gomes, Duarte & Castro, Rui&Henriques, Pedro. (2023). Applying Machine Learning to Estimate the Effort and Duration of Individual Tasks in Software Projects. IEEE Access. PP. 1-1. 10.1109/ACCESS.2023.3307310.

4. Banimustafa, Ahmed. (2018). Predicting Software Effort Estimation Using Machine Learning Techniques. 249-256. 10.1109/CSIT.2018.8486222.

5. Varshini, Priya&Kumari K, Anitha&Janani, D & .S, Soundariya. (2021). Comparative analysis of Machine learning and Deep learning algorithms for Software Effort Estimation. Journal of Physics: Conference Series. 1767. 012019. 10.1088/1742-6596/1767/1/012019.

6. Sánchez, Eduardo &Santacruz, Eduardo &Maceda, Humberto. (2023). Effort and Cost Estimation Using Decision Tree Techniques and Story Points in Agile Software Development. Mathematics. 11. 1477. 10.3390/math11061477.

7. Alsaadi, Bashaer&Saeedi, Kawther. (2022). Data-driven effort estimation techniques of agile user stories: a systematic literature review. Artificial Intelligence Review. 55. 10.1007/s10462-021-10132-x.

8. Vijayan, Akila&Vasuki, A. & JOHNVICTOR, ANITA CHRISTALINE &Ranganathan, Sathiya& Rishi, Priti& Edward, Shirly. (2023). Enhancing Software Testing with Machine Learning Techniques. 329-333. 10.1109/ICSCDS56580.2023.10105028.

9. Kocaguneli, Ekrem&Tosun, Ayse&Basar, Ayse. (2010). AI-Based Models for Software Effort Estimation. 323-326. 10.1109/SEAA.2010.19.

10. Yalçıner, Burcu&Özdeş, Merve. (2019). Software Defect Estimation Using Machine Learning Algorithms. 487-491. 10.1109/UBMK.2019.8907149.