IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

ENHANCEMENT OF DECISION TREE FOR SOFTWARE COST ESTIMATION

[#1] B. Manichandana, [#2] M. laya, [#3] K. Dhathri Guptha, [#4] V. Venkataiah

[1,2,3] *UG Student, Department of CSE, CMR College of Engineering & Technology, Hyderabad, Telangana*

[4] Associate Professor, Department of CSE, &Additional controller of examination CMR College of Engineering &Technology, Hyderabad, Telangana

Abstract: Predicting the amount of effort needed for software development is a major challenge encountered by the software industry. It includes planning, supervising projects, analysing the viability of the system development, preparing proposals and proposals presentation to clients, and among this estimation of the effort for planning is one of the most critical responsibilities. It is necessary to have good effort estimation to conduct a well budget. Accurate software project effort estimation is crucial for the competitiveness and success of software companies. For the forecasting of software program attempts, it's far vital to choose the suitable software program attempt estimation techniques. A number of have been proposed like algorithms, non-algorithms, Algorithms over machine learning, and so on. Generally, Decision Trees have drawn the attention of researchers and changed the direction of Effort Estimation towards computational intelligence. We are utilizing the DT concept in our project, which is a straightforward but effective strategy that serves as the foundation for Random Forest, also referred to as a collection of decision trees. Decision Trees facing data overfitting problem. To overcome this problem enhancement of the DT with the ensemble method proposed in this project and also to evaluate the performance of this proposed method using the COCOMO 81 dataset. To evaluate metrics like MSE and RMSE. Research comparing different methodologies indicates that the proposed technique outperforms prior approaches in terms of results.

Index Terms **- Software Cost Estimation (SCE), Decision Tree (DT), AdaBoost, Pruning, COCOMO 81, Mean Square Error (MRE), Root Mean Square Error (RMSE), and Machine Learning (ML).**

I. INTRODUCTION

SCE holds significant importance in system production cycles, guiding project planning and resource allocation. It involves predicting the approximate cost before software development commences, utilizing mathematical algorithms known as cost estimation models. Despite numerous methodologies proposed, achieving complete accuracy remains a challenge due to the uncertainty inherent in early-stage project inception [1]. Accurate estimations are crucial for effective decision-making and successful project management, preventing budget overruns and delays.Given the paramount importance of software cost estimation, many organizations prioritize its precision. Various advanced machine learning techniques, such as Gradient Boosting, Neural Networks, and Random Forest, are employed for this purpose. Decision Trees, renowned for their simplicity, interpretability, and adaptability to diverse data types, particularly excel in this domain [2]. They effectively capture non-linear relationships and facilitate easy visualization, making them invaluable for software cost prediction insights.This paper introduces an enhanced decision tree method tailored to improve the accuracy of software cost estimation, particularly addressing challenges associated with limited project information during early development stages. Through a comparative analysis with BASIC COCOMO, the proposed approach demonstrates superior effectiveness in achieving refined and reliable cost estimations.

II. BACKWORD WORK

Software cost estimation is a critical process in project management, involving the prediction of resources, time, and budget required for the development of a software project. It aims to provide stakeholders with a realistic assessment of the project's financial and temporal requirements. Several factors influence software cost estimation, including project size, complexity, and requirements clarity. Metrics like lines of code, function points, and use case points are commonly used to quantify the size and functionality of the software [3]. The experience and expertise of the development team, as well as the technology and tools employed, play a significant role in determining costs. Risk assessment is crucial, considering identified risks and mitigation strategies. External factors such as regulatory compliance, security requirements, and dependencies on third-party integrations can introduce additional complexities. Indeed, with the growing significance of software systems, estimating software costs accurately has become increasingly vital. Researchers continually strive to address this challenge by introducing new methods each year. Leveraging machine learning algorithms, some modern approaches aim to enhance the precision of cost estimates. These methods represent a promising avenue for improving software cost estimation in response to the evolving nature of modern software systems.

Reza Ahamad and team from Hacettepe University employed the bee colony optimization (BCO) algorithm to refine cost estimation accuracy based on intermediate COCOMO. By categorizing NASA information set projects according to COCOMO assignment types and calibrating recovery parameters through BCO, they achieved enhanced accuracy. Results indicated a significant improvement, with MMARE declining from 0.2371 percent (COCOMO) to 0.1619 percent (proposed method), reflecting a decrease of 0.0762 percent [4].

Abdel Karri Baareh from Al-Balqa Applied University, Ajloun, Jordan. In his paper, He used two NN models, namely the Back-propagation algorithm and the Radial Base algorithm, to compare their effectiveness in software effort and cost estimation. Using a NASA public dataset, the models are implemented, with 45 data samples for training and 15 for testing. The results indicate that the Back-propagation neural network outperforms the Radial Base function in both training and testing cases [5].

Azeeh Ghatasheh and his team used the Firefly Algorithm to optimize parameters in three COCOMO based software effort estimation models. Experimental results demonstrate the algorithm's high accuracy and substantial error minimization compared to other optimization methods. They concluded that the Firefly Algorithm's superior performance [6].

Mohammad Masdari from Iran University presents a paper on estimating software development project costs using a blend of Genetic Algorithm (GA) and Ant Colony Optimization (ACO), specifically tailored for NASA software projects. The model employs GA for testing and ACO for training, showing enhanced results compared to the conventional COCOMO model. The model's effectiveness stems from its consideration of influential factors in estimation [7].

III. EMPPLOYED TECHNIQUES

The main technique used to find the accuracy of SCE is the DT.

A. Decision Tree

A decision tree visually represents decision-making processes within software, serving as a graphical depiction of these processes. Illustrating potential outcomes and the choices leading to them. Widely utilized across machine learning, business, and decision analysis domains, Decision trees play a crucial role in tasks such as classification and regression [8]. These trees are constructed by iteratively segmenting records based on features to form homogeneous subsets. At each node, the algorithm identifies the most informative feature for data splitting, aiming to maximize information gain or minimize impurity. This iterative process persists until a predefined stopping criterion is met, resulting in a finalized structure that can be applied to new data. Decision trees offer numerous benefits, including interpretability, ease of visualization, and the capability to handle both categorical and numerical data.

B. Overfitting

It is in device studying happens whilst a version learns the education information too well, shooting noise and random fluctuations. This leads to poor generalization, where the model may perform exceptionally on the training set but fails to generalize effectively to new, unseen data. One common technique to address overfitting, especially in decision tree based models, is pruning.

C. Pruning

It entails pruning branches from a decision tree that don't significantly enhance its predictive capability. In the realm of decision trees, overfitting frequently leads to excessively intricate trees that capture the noise present in the training data, rather than the underlying patterns. Pruning aims to simplify the tree, eliminating unnecessary

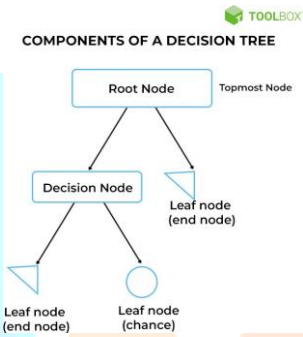
branches and nodes. This process helps prevent the tree from memorizing specific details of the training data that do not represent true patterns, ensuring a more generalized and robust model.

D. Boosting

It's a method in which multiple weak learners are combined to form a robust learner. In the context of decision trees, boosting entails training a sequence of decision trees, with each subsequent tree assigning higher weight to instances misclassified by the preceding trees. The boosting process helps improve the overall performance of the model by iteratively correcting errors made by previous weak learners. Popular boosting algorithms that can be applied to decision trees include AdaBoost and Gradient Boosting (e.g., XGBoost, LightGBM, and CatBoost).

E. AdaBoost

It helps prevent overfitting by paying extra attention to the wrongly guessed examples and forming a strong team of decision trees. It works well with tricky data, gets better at making predictions overall, and is good at handling



noisy information. AdaBoost's special attention to tricky cases makes decision tree models perform better.

Structure of Decision Tree:[11]

To find the accuracy of a software cost estimation, the evaluating Metrics used, are Root Mean Square Error (RMSE) and Mean Square Error (MSE).

B. Error of Root Mean Square (RMSE):

RMSE is the square root of the average squared difference between the predicted values and the actual values in a dataset. The decrease in the RMSE, the higher the version suits a dataset. It is a commonly used metric for evaluating the accuracy of a predictive model, particularly in the context of regression analysis. It measures the common significance of the mistakes among anticipated values and real values in a dataset. RMSE is a popular choice because it penalizes larger errors more heavily than smaller ones, making it sensitive to outliers.[9].

It is calculated as **RMSE = √ 1/n*Σ (ŷi – yi)2**

where: Σ is a symbol that means "sum", ŷi is the predicted value for the ith observation, yi is the observed value for the ith observation and n is the sample size.

C. Square root of mean error (MSE):

MSE quantifies the average squared difference between the predicted values and the actual values in a dataset. The decrease in the MSE, the higher the version suits a dataset [10].

The formula for MSE is **MSE = 1/n * Σ (ŷi – yi)2**

where: Σ is a symbol that means "sum", ŷi is the predicted value for the ith observation, yi is the observed value for the ith observation and n is the sample size.

IV. PROPOSED METHODOLOGY

In this paper, the COCOMO 81 data set is collected from the literature which includes attributes related to software projects, such as reliability, data complexity, time constraints, storage requirements, and numerous additional influencing elementssoftware development efforts and cost after that min max scaler is applied to normalize the dataset features, ensuring they fall within a uniform range of 0 to 1. This step is crucial to prevent certain features from disproportionately influencing the machine learning model due to varying scales. Training and testing portions of the dataset are separated. sets using the train_test_split function. This separation is fundamental for training the model on one subset and evaluating its performance on another, unseen subset. A Tree of Decisions Regressor is trained on the training set. It learns patterns and relationships within the data. Algorithm for decision trees:

- Establish node N for software cost estimation.
- If all projects in dataset D share the same cost estimation class, provide back N as a leaf node with that class labelled.
- If the attribute list was empty: -Return N as a leaf node that has the label average or median cost in D.

- Apply Attribute_selection_method (D, attribute_list) to determine the "best" splitting criterion.
- Tag node N with the selected splitting criterion.
- If the the characteristic that separates is discrete-valued and multiway splits are permitted: >Remove the the characteristic that separates from the attribute lists.

>For each outcome of the splitting criterion:

-Let D_partition be the subset of data projects in D satisfying the outcome.

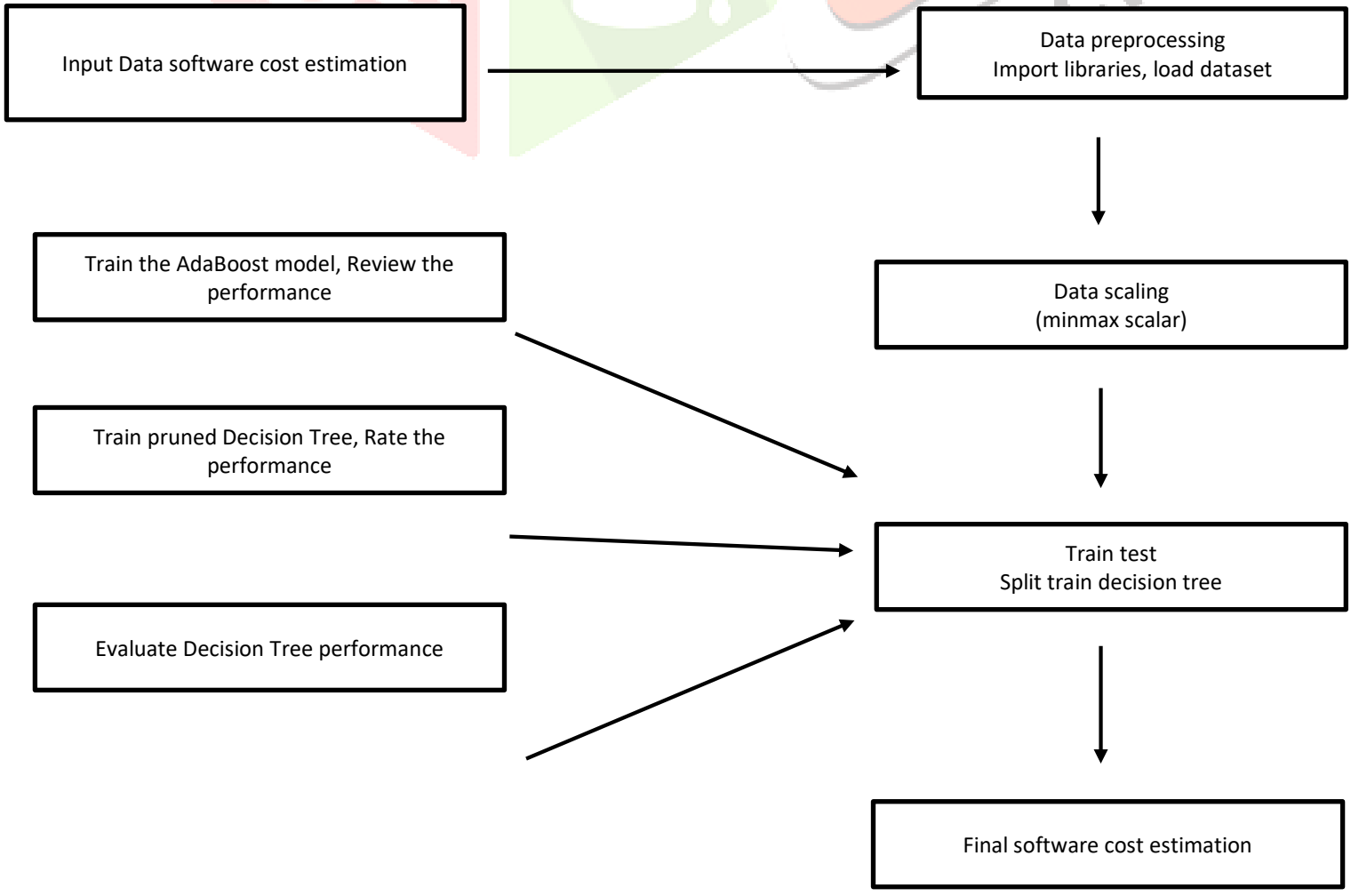
- If D_partition is empty:

-Attach a leaf labeled with the average or median cost in D to node N.

- Else: Attach the node returned by generate_cost_estimation_tree(D_partition,attribute_li st) to node N.
- Return N.

To assess the effectiveness of the Decision Tree model, predictions are made on the test set, and Relevant metrics such as root-mean-squared error (RMSE) and the mean squared error (MSE) are computed. Next, an AdaBoost Regressor is trained on the training set, utilizing the Decision Tree Regressor as a base estimator. AdaBoost blends a number of ineffective learners to create a stronger predictive model. The AdaBoost model's performance is then assessed by making predictions on the test set and computing performance metrics such as MSE and RMSE. Following that, a pruned Decision Tree Regressor with limited depth is trained on the training set. Pruning helps prevent overfitting and simplifies the model. The performance of the pruned Decision Tree is evaluated on the test set, and performance metrics (MSE and RMSE) are calculated. The final software cost estimation output showcases the error evaluation before and after the enhancement of the decision tree model, specifically addressing the reduction of overfitting. This analysis demonstrates how the Model of decision trees is refined and improved, ultimately enhancing its predictive capabilities for software cost estimation.

BLOCK DIAGRAM



IV. RESULTS AND DISCUSSION

In this document, the software cost estimation results demonstrate notable improvements after applying enhancement techniques such as AdaBoost and pruning. Before enhancement, the RMSE stood at 0.05102516893992321, while the MSE was 0.0026035678653477047. Following the application of AdaBoost, the RMSE decreased to 0.04352386357990304, and the MSE reduced to 0.0018943267009220099. Pruning, another enhancement method, resulted in an RMSE of 0.05004060795782995 and MSE of 0.002504062444789234. Both techniques contributed to a decrease in errors, with AdaBoost exhibiting a more pronounced effect. The choice between AdaBoost and pruning should consider factors beyond error reduction, such as computational complexity and the specific objectives of the SCE.

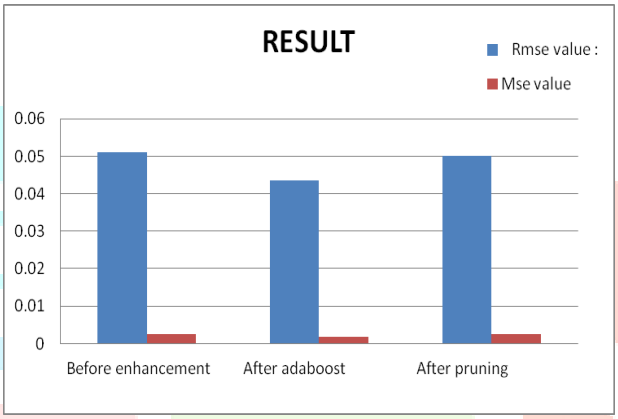


Fig.1: Result analysis

V. CONCLUSION

In today's software development landscape, accurately estimating the costs of complex systems early on is crucial, given the rising expenses associated with software projects. However, achieving perfect prediction accuracy remains an elusive goal in software engineering research. This work tackles the problem by putting forth a strategy based on the fundamental COCOMO model that uses decision tree algorithms to improve cost estimation accuracy. Through the application of Decision Tree models alongside AdaBoost and pruning techniques on the COCOMO 81 dataset, the study demonstrates significant improvements in accuracy. AdaBoost is especially notable for its significant improvement, although pruning provides a more sophisticated method. The results showcase a noticeable uptick in cost estimation precision, with the RMSE dropping from 0.051 percent to 0.043 percent after implementing AdaBoost and pruning.

REFERENCES

[1] International Journal of Computer Applications (0975 – 8887) Volume 104 – No 12, October 2014

[2]J.Caper, "Estimating Software Costs (English) 2nd Edition", Tata McGraw - Hill Education, 2007.

[3] Z. A. Khalifehlou, F. S. Gharehchopogh, "A Survey of Data Mining Techniques in Software Cost Estimation", AWER
Procedia Information Technology & Computer Science Journal, Vol:1, pp.331- 342, 2012

[4] Z.A. Khalifelu, F.S. Gharehchopogh, "Comparison and Evaluation Data Mining Techniques with Algorithmic Models in
Software Cost Estimation", Elsevier Press, Procedia-Technology Journal, ISSN: 2212-0173, Vol: 1, pp. 65-71, 2012.

[5] Abdel Karim Baareh / Journal of Computer Science 2019, 15 (3): 321.331 DOI: 10.3844/jcssp.2019.321.331

[6] Baareh, A. K. (2019). Optimizing Software Effort Estimation Models Using Back-Propagation Versus Radial Base
Function Networks. Journal of Computer Science, 15(3), 321-331. https://doi.org/10.3844/jcssp.2019.321.331

[7] International Journal of Innovation and Applied Studies ISSN 2028-9324 Vol. 5 No. 1 Jan. 2014, pp. 72-81 © 2014

Innovative Space of Scientific Research Journals http://www.issr-journals.org/ijias/

[8] Aitkenhead MJ (2008) A co-evolving decision tree classification method. Exp Syst Appl 34: 18–25

[9] T. Chai1,2 and R. R. Draxler1 1NOAA Air Resources Laboratory (ARL), NOAA Center for Weather and Climate

Prediction, 5830 University Research Court, College Park, MD 20740, USA

[10] Allan H. Murphy Department of Atmospheric Sciences, Oregon State University, Corvallis, Oregon DOI: https://doi.org/10.1175/15200493(1988)116%3C2417:SSBOT M%3E2.0.CO;2

[11] https://www.spiceworks.com/tech/artificial- intelligence/articles/what-is-decision-tree/