Heterogeneous Ensemble Model For Software Effort Estimation

1st Sai Kamal Gaddala
Computer Science and Engineering
National Institute of Technology
Warangal,India
gs21csb0a17@student.nitw.ac.in

2rd Pranay Suhas Elkapelly
Computer Science and Engineering
National Institute of Technology
Warangal,India
ep21csb0a16@student.nitw.ac.in

3th Sanjay Vadithya
Computer Science and Engineering
National Institute of Technology
Warangal,India
vs21csb0a64@student.nitw.ac.in

4th Manjubala Bisi
Computer Science and Engineering
National Institute of Technology
Warangal,India
manjubalabisi@nitw.ac.in

5th Sarika M
Computer Science and Engineering
National Institute of Technology
Warangal,India
ms23csr1r02@student.nitw.ac.in

***Abstract*—Software projects are crucial part of a company. To manage software projects, companies need to know the effort required to complete software projects. By estimating the effort of a software project, a company can manage resources accordingly. Various standalone models are used before to estimate the effort of software projects. This study takes a step forward from the standalone models and combines different standalone models to create a single ensemble model. To test our model, we compared the results of standalone models to the ensemble model for benchmark datasets.**

***Index Terms*—Software Effort Estimation, Standalone model, Heterogeneous ensemble model**

I. Introduction

Estimation of software effort plays a crucial role in project management by predicting the time, cost, and resources required for successful project completion. Accurate estimation helps organizations optimize resource allocation, minimize risks, and improve project planning.Traditional approaches like Expert Judgement, Delphi Assessments, Algorithmic models (e.g., COCOMO, Function Point Analysis) and machine learning approaches (e.g., Neural Networks,Regression Models) face challenges such as data variability, noise, and the complexities of project dynamics.

To address these issues, a heterogeneous ensemble model has proven to be helpful. Unlike independent models, which use one approach to make predictions, heterogeneous ensemble models employ various estimation methods to take advantage of each and counteract their respective weaknesses.

This study focuses to give better and more accurate estimations for decision making in software project management by combining several methodologies. By exploring various ensemble strategies and evaluating their performance against traditional and standalone models, this research aims to demonstrate the effectiveness of a multi-model approach in refining effort estimation.

II. Literature Review

Software effort estimation is an important part of a software project, so various methods have been proposed until now to estimate the software effort. Various methods such as expert judgment, algorithmic models, and machine learning models are proposed.

Expert judgement is one of the earliest methods for estimating software effort. This method works by consulting experts like professionals and project managers to predict the effort estimation for a software project. Experts predict the software effort using their past experience. They compare the given software project for estimation with previous similar software projects.

Algorithmic models use mathematical equations to estimate the effort for a software project. The mathematical equations consist of different parameters of project such as size, complexity etc. which directly effect the effort. To predict the effort, these models depend on statistical relationships.

Machine learning models uses the historical data to train model, this model learns patterns among the data and uses them to estimate the software effort for new projects.

Every method has its own pros and cons, so each model can be used accordingly, but the model that accurately estimates the software effort is required. So we proposed a model called heterogeneous ensemble model, which combines different standalone models and uses the results of the each model to estimate the final software effort. By doing this strengths of models can be combined which yields in better accuracy in estimating software effort.

III. Methodology

This section provides the methodology used for developing heterogeneous ensemble model. We used CBR, COCOMO II, XGBoost standalone models. We define standalone models and heterogeneous ensemble model in this section.

A. Standalone Models

These are the individual models used in the ensembling process.

*1) **Case-Based Reasoning (CBR)**:* Case-Based Reasoning (CBR) is a reasoning technique that solves new problems based on historical data, i.e., by referencing similar past cases. CBR assumes that similar projects have similar software development efforts. This method is widely used in Software Effort Estimation as it uses historical project data for predicting the effort of new projects.

Process of Effort Estimation: CBR involves four major phases to predict effort. Each of these phases are explained below:

1) **Case Retrieval:** This phase involves retrieving the most relevant past cases. The new project attributes (e.g., size, complexity, development time) are compared with past projects stored in a case base. The similarity between two cases is measured using distance metrics such as Euclidean distance, cosine similarity, or weighted similarity functions.
2) **Case Reuse:** The retrieved past cases are analyzed, and their effort estimates are adjusted to better fit the attributes of the new project. These modifications consider variations in the project attributes.
3) **Case Revision:** If needed, specialists refine the estimate by incorporating project-specific factors.
4) **Case Retention:** After the project's completion, the actual effort is recorded in the case base for future use.

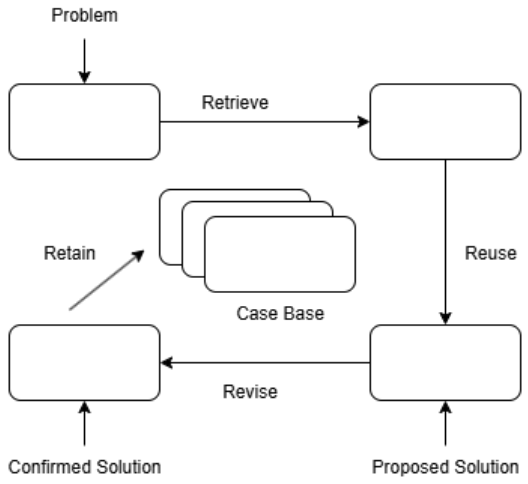

Fig. 1. Case Based Reasoning(CBR)

The major advantages of CBR is its ability to handle uncertainties in software effort estimation and its adaptability to new data. However, the effectiveness of CBR depends on the quality of stored cases and the choice of similarity measurement.

*2) **COCOMO II (Constructive Cost Model)**:* COCOMO II is an algorithmic cost estimation model that predicts the effort required for software development based on project size and various cost drivers. It adjusts effort estimates using 17 cost drivers, each having a predefined rating (e.g., Very Low to Extra High), influencing the final effort estimate.

1) **Estimation Formula:** The equation for estimating effort in person-months is:

$$\text{Effort} = A \times (\text{Size})^B \times \prod E_i \tag{1}$$

Where:

- A, B - Model-specific constants.
- Size - Estimated code size in KLOC (Kilo Lines of Code).
- E_i - Effort multipliers based on cost drivers (e.g., complexity, team experience).

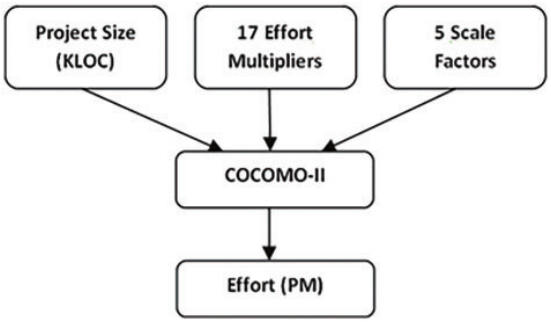

Fig. 2. COCOMO II

COCOMO-II have some major advantages like this is applicable to different software development stages and supports modern software development paradigms like object-oriented and agile approaches.

*3) **XGBoost (Extreme Gradient Boosting)**:* XGBoost is a powerful and efficient machine learning approach. Basically,it is a gradient boosting algorithm that builds an ensemble of decision trees sequentially. Each tree learns from the errors of the previous trees to minimize prediction loss. XGBoost has a Boosting process where it starts with a weak model(a small decision tree) and then it gradually improves by reducing the residual errors of previous decision trees.To optimize the performance gradient descent approach is used in boosting process.Internally it also uses Regularization Techniques like L1(Lasso) and L2(Ridge) regularisation. Along with this, it also uses maximum depth constraints to avoid overfitting.

XGBoost works initially by making a prediction of taking the average of the target values, and then compute the difference (i.e., the residual error) between actual values and predictions. In order to improve accuracy, new decision trees are built with each one focused on correcting errors made by previous ones. This continues iteratively, further improving predictions. The major advantages of XGBoost is that it capture complex relationships providing high accuracy and also handles the missing data very well compared to other machine learning models.Also, its fast computing made possible by parallel processing greatly accelerates training,which makes it perfect for managing large amounts of data.

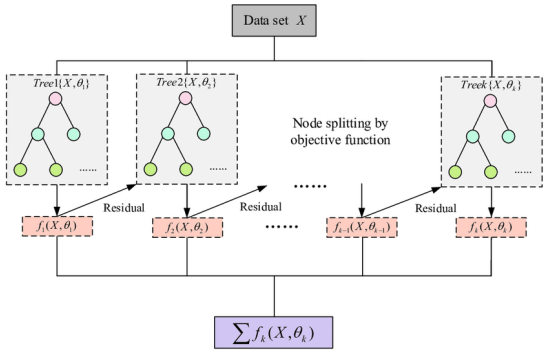

Fig. 3. XGBoost (Extreme Gradient Boosting)

Furthermore, XGboost has different important hyperparameters,affecting the performance of the model. Some include learning rate, max-depth, Number of Trees (n-estimators). These hyperparameters should be properly tuned to get the best performance.

B. Heterogenous Ensemble Model

After the standalone models have been assessed, a heterogeneous ensemble model was modelled to combine the strengths of multiple different types of models. This ensemble model involves three individual models:

- **COCOMO-II**: A parametric model which estimate the effort based on project size and scale factors.
- **Case-Based Reasoning (CBR)**: A memory based method that estimates effort by retrieving and adapting from the similar past projects.
- **A Machine Learning Regressor**: One of SVR, Linear Regression, K-NN, ANN, or XGBoost is choosed.

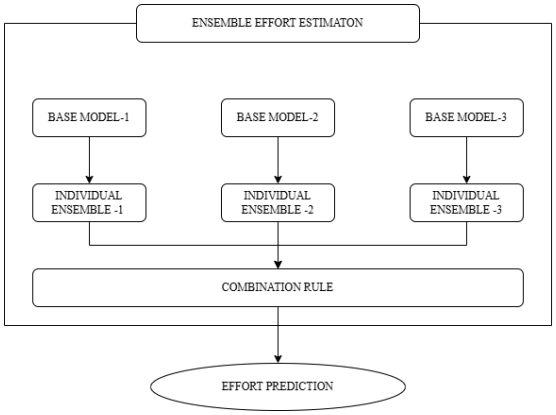

Fig. 4. Heterogeneous Ensemble Model

The purpose of the ensemble method is to take advantage of the distinctive characteristics of each model, interpretability of algorithmic approaches (COCOMO-II) and the adaptability of machine learning models. They are trained independently on the dataset, and their results (effort estimates) are subsequently combined with a combination rule to create a final effort estimation.

This technique is categorized as heterogeneous since it com bines individually different type of models rather than using a multiple homogeneous models. Ensemble was tested with multiple data sets to verify its stability and generalization.

C. Combination Rule

To combine the predictions from the three models (COCOMO-II, CBR, and the choosed machine learning model) different combination rules were applied. These rules define how the individual predictions are merged to get a final effort estimation:

- **Linear Combination Rule:** In this approach, each output of a model is assigned with a weight, and the final prediction is predicted as the weighted sum of all three outputs of Individual Models. These weights can also be adjusted to increase the performance of model.

$$E_{\text{final}} = w_1 \cdot E_{\text{CBR}} + w_2 \cdot E_{\text{Cocomo}} + w_3 \cdot E_{\text{ML-approach}} \quad (2)$$

$$w_1 + w_2 + w_3 = 1 \quad (3)$$

- **Median Combination Rule:** In this approach, the median of the individual predictions is taken as the final prediction. This is useful when the model can produce outlier values, as the median is powerful for handling extreme values.

$$E_{\text{final}} = \text{median}(E_{\text{Cocomo}}, E_{\text{CBR}}, E_{\text{ML-approach}}) \quad (4)$$

Each combination rule was evaluated to determine the best combination rule. Each of these rules have advantages like the linear combination allows for controlled contribution from each model and the median rule offers robustness to handle inconsistent predictions.

IV. Results

This section provides a comparative analysis of the proposed Ensemble model in relation to individual models with different configurations. To calculate accuracy, we have used performance metrics that include MAE, MMRE, MDMRE, and PRED(0.25). The performance evaluation was conducted using three widely recognized software effort estimation datasets NASA60,NASA93 and COCOMO81.

A. Performance of Individual Standalone Models

Based on the experimental results shown in figure 4 and 5,XGBoost Regressor showed the optimal balance between accuracy and training times as shown in the figure.While ANN achieved the lowest MAE,but its significantly higher training times made it less practical with large datasets like Cocomo81.On the other hand, XGBoost was a more effective and scalable option because it consistently produced low MAE across all datasets while keeping training times manageable. While Linear Regression, even though it was simple, did not do well, especially on the COCOMO81 dataset, showing that it is not good at capturing complex effort estimate patterns.Similarly KNN and SVR did not perform well when compared with XGBoost.

In conclusion, XGBooster emerged as the best choice for Ensembling with CBR and cocomoII due to ability to handle feature interactions,stop overfitting, and optimize computation through parallel processing. Thus, considering both accuracy and computational times, XGBoost was chosen as the final model for software effort estimation.

TABLE I
PERFORMANCE OF VARIOUS STAND-ALONE MODELS FOR NASA60

Models	Training Time (s)	MMRE	MAE	MDMRE	PRED(0.25)
SVR	0.1604	0.9163	265.5283	0.3960	0.3667
Linear Regression	0.1159	1.7822	249.3008	0.6457	0.2500
K-NN	0.1003	0.4803	159.8280	0.2873	0.4333
XGBoost Regressor	10.8870	0.3105	155.4475	0.1970	0.6000
ANN	72.0850	0.4069	144.6046	0.2546	0.4833
CBR	0.0031	0.4381	173.1817	0.3470	0.3833
COCOMO II	0.0002	0.2539	96.6304	0.1977	0.6500

TABLE II
PERFORMANCE OF VARIOUS STAND-ALONE MODELS FOR NASA93

Models	Training Time (s)	MMRE	MAE	MDMRE	PRED(0.25)
SVR	0.6864	0.9642	456.1414	0.5740	0.3011
Linear Regression	0.4145	2.6187	469.7880	0.6796	0.1505
K-NN	0.2912	1.5446	494.1114	0.7156	0.1720
XGBoost Regressor	15.9983	1.1594	395.8388	0.3580	0.3656
ANN	21.5327	1.5407	345.4534	0.7670	0.2258
CBR	0.0050	1.6340	571.9526	0.6962	0.2043
COCOMO II	0.0003	0.5937	470.0133	0.2709	0.4624

TABLE III
PERFORMANCE OF VARIOUS STAND-ALONE MODELS FOR COCOMO81

Models	Training Time (s)	MMRE	MAE	MDMRE	PRED(0.25)
SVR	0.1674	1.7206	612.2693	0.8245	0.1746
Linear Regression	0.1773	20.0269	958.4834	5.1893	0.0952
K-NN	0.0847	1.8015	580.0070	0.7481	0.0794
XGBoost Regressor	12.4247	1.3805	319.1951	0.7049	0.2063
ANN	85.9392	5.4754	687.9563	1.5318	0.1587
CBR	0.0032	1.8077	648.9275	0.7725	0.1111
COCOMO II	0.0002	0.3811	219.6751	0.2976	0.4286

B. Performance of Ensemble Model

In order to analyze how effectiveness of proposed model in estimating software effort, we developed and experimented with five combinations. Each of these combinations comprised COCOMO-II, Case-Based Reasoning (CBR), and a single machine learning approach - SVR, Linear Regression, K-NN, XGBoost Regressor, ANN. These are tested on three benchmark datasets—NASA93, COCOMO-II and NASA60 using performance metrics such as MAE (Mean Absolute Error), MMRE (Mean Magnitude of Relative Error), MDMRE (Median of MRE), and PRED(0.25). The goal was to determine which machine learning model improves the predictive power of the COCOMO-II and CBR models. The results are shown in the following Tables.

TABLE IV
PERFORMANCE OF VARIOUS LEARNING MODELS IN CBR-COCOMOII-BASED ENSEMBLE FOR NASA60

Models	Training Time (s)	MMRE	MAE	MDMRE	PRED(0.25)
SVR	0.1637	0.3259	144.8552	0.2799	0.4667
Linear Regression	0.1193	0.3310	114.4340	0.2145	0.6167
K-NN	0.1036	0.3592	133.0375	0.2411	0.5167
XGBoost Regressor	10.8903	0.2853	129.8469	0.1937	0.5833
ANN	72.0883	0.2829	110.7127	0.2073	0.6000

TABLE V
PERFORMANCE OF VARIOUS LEARNING MODELS IN CBR-COCOMOII-BASED ENSEMBLE FOR NASA93

Models	Training Time (s)	MMRE	MAE	MDMRE	PRED(0.25)
SVR	0.6917	0.6844	365.7587	0.4120	0.3656
Linear Regression	0.4197	0.7252	312.4596	0.3939	0.3011
K-NN	0.2965	1.0735	428.1312	0.5643	0.2473
XGBoost Regressor	15.0036	0.7680	339.5386	0.3467	0.3763
ANN	21.5380	0.8504	321.7353	0.4929	0.3011

TABLE VI
PERFORMANCE OF VARIOUS LEARNING MODELS IN CBR-COCOMOII-BASED ENSEMBLE FOR COCOMO81

Models	Training Time (s)	MMRE	MAE	MDMRE	PRED(0.25)
SVR	0.1708	1.1494	387.4852	0.6164	0.2063
Linear Regression	0.1807	0.7410	345.5308	0.3499	0.3651
K-NN	0.0881	1.4325	507.7392	0.6912	0.1587
XGBoost Regressor	12.4282	0.8143	284.4057	0.3853	0.2540
ANN	85.9426	0.9575	410.3751	0.3620	0.3333

The proposed CBR-COCOMOII-based ensemble process was reported to have competitive performance when used with various learning models on NASA60, NASA93 and COCOMO81 datasets. The ANN and XGBoost Regressor consistently reported lower MMRE and MAE values, indicating higher estimation accuracy, especially on NASA60, but the training times of ANN is higher compared to XGBoost Regressor for most of the datasets. Linear Regression also reported good performance on MAE and PRED(0.25), while SVR and K-NN reported higher error rates on larger datasets like NASA93 and COCOMO81. Overall, the results indicate the versatility of the ensemble, and ANN and XGBoost were reported to be promising models to improve estimation precision in the CBR-COCOMOII framework.

C. Comparison Of Ensemble model with stand-alone models

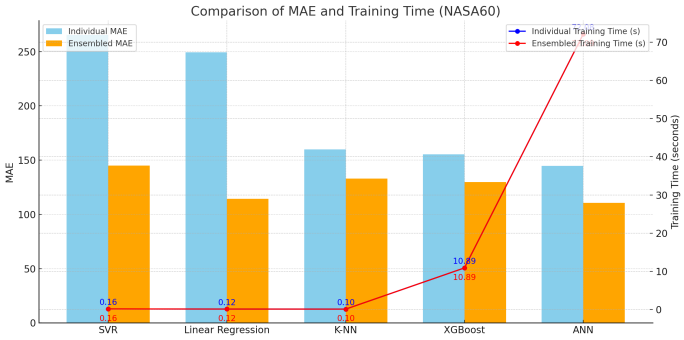

Fig. 5. Comparison of MAE and Training Time (NASA60).

To validate the heterogeneous ensemble model, we compared the results of the stand-alone models to the proposed heterogeneous ensemble model. Metrics such as MAE, MMRE, MDMRE, PRED(0.25) for stand-alone models of ml are compared to heterogeneous ensemble model which is a stand-alone model combined with CBR and COCOMO II. By the results we can say the heterogeneous ensemble model which contains a ml model performs better compared to that particular ml model used individually. The ensemble model produces

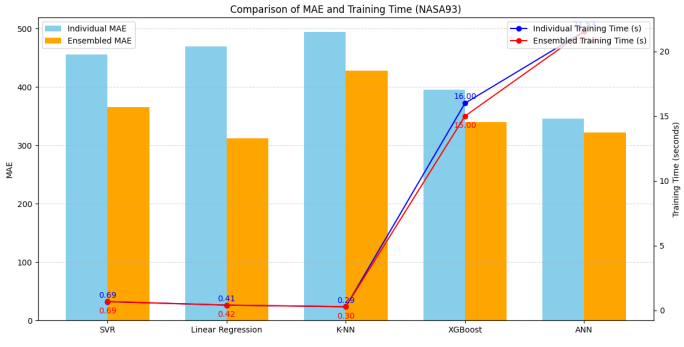

Fig. 6. Comparison of MAE and Training Time (NASA93).

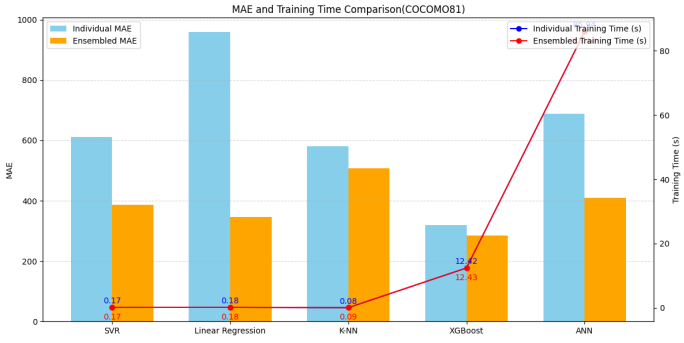

Fig. 7. Comparison of MAE and Training Time (COCOMO81).

better results than stand-alone model without any significant change in the training time. Results are compared for various datasets, the results remain same that is the ensemble model performs better than stand-alone model. By this we can say that the heterogeneous ensemble model outperforms stand-alone model.

V. Conclusion

This study proposed a Heterogeneous Ensemble approach to estimate the effort required to complete a project by integrating Cocomo-II,Case based reasoning(CBR) and a machine learning model like SVR,Linear Regression,KNN,ANN and XGBoost Regressor.After the execution of standalone models, Ensemble method was modelled to increase the strength of individual models by combining the properties like expert knowledge, historical similarity, and data-driven learning.

The ensemble model was also tested with two types of combination rules.Results showed that median combination is better option compared to linear combination as it is more sensitive to the weight distribution.

Among all combinations tested, the COCOMO-II + CBR + XGBoost ensemble, using the median combination rule performed better than other configurations across all datasets.These finding helps in increasing the use of Heterogeneous Ensemble models as a solution to estimate software effort estimation in real-world project management.

References

[1] Ali, Syed Sarmad, Jian Ren, Kui Zhang, Ji Wu, and Chao Liu. "Heterogeneous ensemble model to optimize software effort estimation accuracy." IEEE Access 11 (2023): 27759-27792.

[2] M. Z. M. Hazil, M. N. Mahdi, M. S. M. Azmi, L. K. Cheng, A. Yusof, and A. R. Ahmad, "Software project management using machine learning technique—A review," in Proc. 8th Int. Conf. Inf. Technol. Multimedia (ICIMU), Aug. 2020, pp. 363–370.

[3] J.Wen,S.Li,Z.Lin,Y.Hu,andC.Huang,"Systematicliterature review of machine learning based software development effort estimation models," Inf. Softw. Technol., vol. 54, no. 1, pp. 41–59, Jan. 2012.

[4] Idri, Ali, Mohamed Hosni, and Alain Abran. "Systematic literature review of ensemble effort estimation." Journal of Systems and Software 118 (2016): 151-175.

[5] Heemstra, Fred J. "Software cost estimation." Information and software technology 34, no. 10 (1992): 627-639.

[6] Gandomani, Taghi Javdani, Maedeh Dashti, Hazura Zulzalil, and Abu Bakar Md Sultan. "Enhancing Software Effort Estimation in the Analogy-Based Approach Through the Combination of Regression Methods." IEEE Access (2024).

[7] Mukhopadhyay, Tridas, Steven S. Vicinanza, and Michael J. Prietula. "Examining the feasibility of a case-based reasoning model for software effort estimation." MIS quarterly (1992): 155-171.

[8] Sousa, André O., Daniel T. Veloso, Henrique M. Gonçalves, João Pascoal Faria, João Mendes-Moreira, Ricardo Graça, Duarte Gomes, Rui Nuno Castro, and Pedro Castro Henriques. "Applying machine learning to estimate the effort and duration of individual tasks in software projects." IEEE Access 11 (2023): 89933-89946.

[9] B. W. Boehm and R. Valerdi, "Achievements and challenges in cocomo based software resource estimation," IEEE Softw., vol. 25, no. 5, pp. 74–83, Sep. 2008.

[10] D. Wu, J. Li, and C. Bao, "Case-based reasoning with optimized weight derived byparticleswarmoptimizationforsoftwareeffortestimation,"Soft Comput., vol. 22, no. 16, pp. 5299–5310, Aug. 2018.