International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(4s) 2026 pp. 742-753
https://doi.org/10.70917/ijcisim-2026-2562
Article

A Novel AI-Enabled Metaheuristic Optimization Framework for Accurate Software Cost Estimation Using Benchmark and Industrial Datasets

M S Rekha[1*], Bharathi Malakreddy A.[2], Rajesh I S.[3]

[1]Department of Computer Science and Engineering, B M S Institute of Technology &Management, Yelhanka, Affiliated to Visvesvaraya Technological University, Belagavi-590018, Karnataka, India.
*Department of Computer Science and Engineering, R L Jalappa Institute of Technology, Doddaballapur-561203, India. Email ID: rekha061264@gmail.com
[2]Department of Artificial Intelligence and Machine Learning,B M S Institute of Technology &Management, Yelhanka,Karnataka,India. Email ID: bharathi_m@bmsit.in
[3]Department of Artificial Intelligence and Machine Learning,B M S Institute of Technology &Management, Yelhanka,Karnataka,India. Email ID: rajeshaiml@bmsit.in

Abstract: — Accurate software cost estimation is a significant aspect in software project management because it has a direct impact on software projects. However, software cost estimation techniques have some limitations in managing high-dimensional software project variables and nonlinear relationships between software cost drivers. These limitations often lead to inefficient software project management and inaccurate software cost estimation. In order to overcome these limitations and inaccuracies in software cost estimation techniques, this paper proposes a novel AI-based metaheuristic optimization technique for software cost estimation by integrating more accurate machine learning models and intelligent optimization techniques.The proposed software cost estimation model incorporates a Transformer regression model and a Genetic Algorithm (GA) to efficiently estimate software cost. The significant software project characteristics used in this proposed software cost estimation model are project size in KLOC, function points, requirement stability, team size, developer experience, productivity index, system complexity, technology maturity, and tool automation level. Performance of model is measured by common evaluation metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Magnitude of Relative Error (MMRE). The experimental outcome shows that proposed hybrid model has significant effect improving the accuracy of software cost prediction compared to the existing machine learning and estimation techniques. The model shows an accuracy of about 92% with less estimation errors, proving its efficiency in modeling complex project characteristics. The proposed AI-powered metaheuristic optimization model presents a robust and efficient solution for intelligent software cost estimation.

Keywords: — Artificial Intelligence, Software Cost Estimation, Metaheuristic Optimization, Machine Learning, Feature Selection, Prediction Model.

1. Introduction

Software cost estimation is important software engineering activity that has a direct impact on software project planning and budgeting. Software projects are usually characterized by complex interactions among various factors such as project size, system complexity, software developers' skills, requirement stability, and technology used for software development [1]. Conventional software cost estimation models like algorithmic and expert models usually face difficulties in identifying non-linear interactions among various factors and hence produce less accurate results. As a result, software cost estimation accuracy improvement is an important research problem in contemporary software project management [2]. Recently, intelligent prediction models have been widely used to overcome the

shortcomings of traditional software cost estimation models due to rapid advancement of AI and ML technology. Machine learning models can identify complex interactions between various factors using historical software project data and adapt to such interactions accordingly [3]. However, many traditional machine learning techniques also face some challenges in terms of high-dimensional data sets, redundant features, and improper parameter settings. All these factors might affect the generalization ability of the models and further impact the prediction efficiency of the models when implemented in real-world software development environments [4].

However, to overcome the limitations of traditional machine learning techniques, some metaheuristics optimization techniques have been incorporated into the traditional models to improve the feature selection and parameter tuning efficiency of the models [5]. One such optimization technique is the GA, which is a natural evolutionary optimization technique used to find the optimal feature sets and minimize the prediction error of the models. GA is used to select the most important attributes of the projects to improve the efficiency of the models [6]. In the present work, a software cost estimation framework through the integration of a regression model based on the Transformer architecture and a GA optimization method has been proposed. The Transformer architecture relies on attention mechanisms to incorporate the dependencies between the parameters of the software project to accurately estimate the cost of software development [7]. The GA assists in the feature selection and optimization processes to identify the most influencing cost factors from the dataset. The results show accuracy of proposed model compared traditional cost estimation methods. Another important issue to be addressed in software cost estimation is the presence of high-dimensional project attributes and redundant cost drivers in large data sets [8].

If redundant attributes are used during the learning process, it might result in overfitting and increased computational complexity. Therefore, it is important to identify the most contributing project parameters to develop an efficient software cost estimation model. Recently, deep learning models, such as the transformer model, have been effective in modelling complex relationships within a given data set [9]. The transformer model uses the self-attention mechanism to focus on the most important input features and their dependencies. The Transformer model has the advantage of being able to handle more than one feature at once compared to other traditional neural networks. This advantage makes it one of the most efficient models for dealing with complex prediction problems, including software cost estimation, where multiple project attributes are involved in a non-linear relationship. In addition, the incorporation of metaheuristic optimization algorithms into DL models has identified as promising solution for improving the efficiency of the prediction model. This is because the GA has been recognized for its ability to carry out a global optimization search through evolutionary operations such as selection, crossover, and mutation operations [10].

2. Related works

Software cost estimation has been a vital research area of software engineering field for the last few decades. Conventional software cost estimation models, Function Point Analysis, and expert judgment approaches have been broadly used to estimate effort, cost associated with software development projects [11]. These approaches mainly rely on mathematical formulas and project parameters like project size, complexity, and developer experience. Despite the availability of a well-structured estimation methodology, the conventional approaches do not consider the nonlinear relationships between project attributes, which may result in incorrect predictions in large software development projects [12]. To mitigate the disadvantages associated with traditional approaches, various ML approaches have been proposed to estimate the cost associated with software projects. In the field of ML, various algorithms like Linear Regression, SVM, Decision Trees, RF, ANN, etc., have been proposed to estimate the cost associated with software projects based on the available historical data [13]. Several studies have shown that ML techniques can used to improve accuracy of predictions significantly when compared to other algorithmic models using benchmark data sets. In recent times, studies have been carried out to improve estimation accuracy using machine learning models through optimization techniques. Metaheuristics as GA, PSO, and ACO are used to improve estimation accuracy [14].

These algorithms are used to optimize features and parameters to avoid redundant features, which improve the efficiency of machine learning models and minimize prediction errors. Optimization algorithms are used to improve machine learning model estimation accuracy [15]. Apart from the traditional ML models, DL models have also gained recent research attention for software cost estimation problems. DNN, along with their variants, have shown promising results in learning complex interactions between the features of large datasets. Among these deep learning models, the Transformer-based models, employing the attention mechanism, have shown promising results in dealing with multiple input variable relationships [16]. This ability of the deep neural networks is useful in efficiently learning the relationships between the characteristics of the software project, enabling the development of complex prediction

models.However, the existing literature still has limitations, mainly related to dealing with high-dimensional datasets, dealing with redundant cost drivers, and optimizing the parameters of the model. Therefore, a new direction has been proposed, where the deep neural networks are integrated with the metaheuristic optimization algorithms, often referred to as hybrid models. In the context of the proposed framework, a deep neural network model, referred to as the Transformer, has been integrated with a GA optimization model [17].

Additionally, several research studies have attempted to enhance the precision of the estimation process by employing hybrid models of machine learning techniques. In this case, ensemble learning techniques such as RF, Gradient Boosting, and hybrid-NN models are used to develop more accurate software cost estimation systems [18]. These models are effective in enhancing the robustness of software estimation cost system. However, the feature selection process and parameters of the models are critical in enhancing the performance of the ensemble models, especially in handling complex software project data sets.In addition, the availability of large-scale software project data sets has prompted researchers to develop intelligent data-driven software cost estimation models [19]. By leveraging the availability of the software project data sets and employing sophisticated learning models, contemporary software cost estimation models have proven to be effective in identifying hidden relationships between costs of software projects and various characteristics of software projects. However, there is a need to develop adaptive and scalable software cost estimation models to support diverse characteristics and development environments for software projects. In this regard, the integration of deep learning models and metaheuristic optimization algorithms is a promising approach in developing accurate and reliable software cost estimation models [20].

3. METHODOLOGY

The proposed method for cost estimation in software development incorporates artificial intelligence and a metaheuristic optimization technique for precise and accurate cost estimation. Additionally, the proposed method incorporates existing data related to software projects that have been gathered from reliable sources. Moreover, the method incorporates real-time data related to software projects. The data sets include various attributes such as project size in KLOC, function points, requirement stability, team size, developer experience, productivity index, system complexity, technology maturity, and tool automation. Each data point in the sets represents a completed software project with its characteristics and cost. The data is used to develop a predictive model.

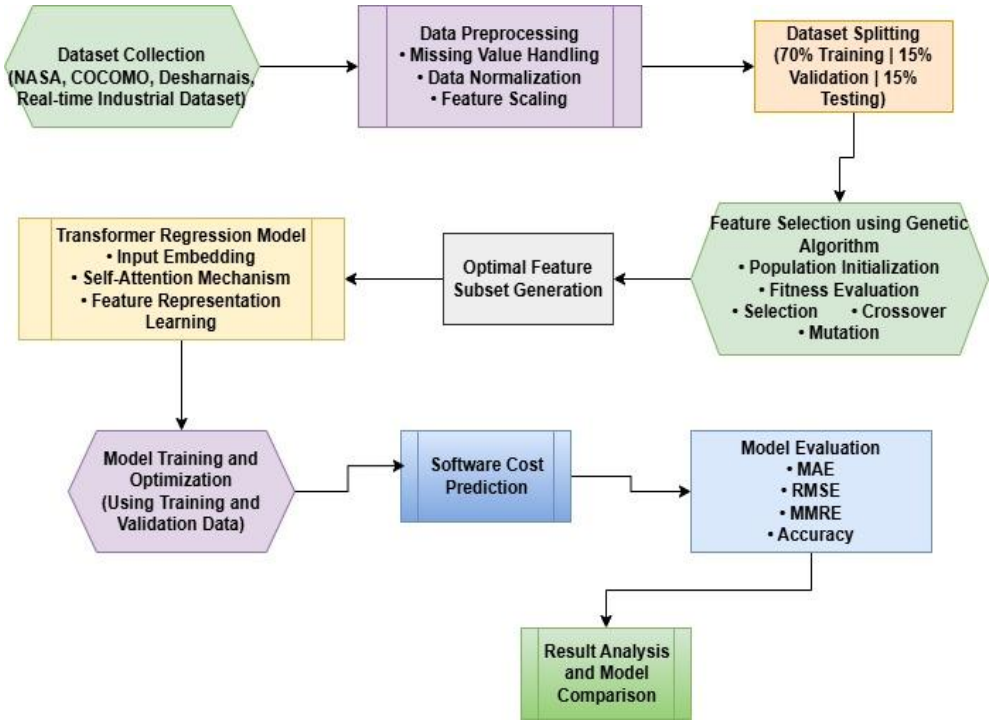

Fig.1 Software Cost Estimation

Fig.1. Illustration of the proposed AI-enabled framework for intelligent software cost prediction. The proposed framework utilizes a combination of metaheuristic optimization and transformer regression for precise cost prediction.

To further boost the efficiency of the prediction process, a GA is applied for feature selection. The GA is metaheuristic optimization technique based on a population search method. The GA is motivated by natural selection and evolutionary processes in nature. The GA is based on the evolutionary process in nature, where selection, crossover, and mutation occur. A chromosome is used in this algorithm as a possible set of features for the software project. The algorithm is further used to find the optimal feature set. This is used to find the most important cost drivers while filtering out irrelevant features. Once the optimal feature set is identified, a transformer regression model is developed for software cost prediction. The Transformer model is based on a self-attention mechanism to identify complex dependencies among software project attributes. The Transformer model is different from a traditional neural network model because it can process multiple features simultaneously. The Transformer model is also able to identify nonlinear relationships between software project attributes. The feature subset determined by the GA is used to train the Transformer model.

The datasets contain the project attributes along with the corresponding development effort, which is used to develop the cost estimation models (1).

$$D = \{(X_i, Y_i)\}_{i=1}^{N} \quad (1)$$

Where,

D –Dataset

X_i – Input feature

Y_i – Development cost or effort

N – No of projects

Data preprocessing used to improve quality of dataset. This step is useful to handle the missing values, normalize the attributes, and provide the structured input to the model. This step improves the stability of the learning process as well as the bias of the model(2).

$$X' = \frac{X-M}{A} \quad (2)$$

Where,

X' – Normalized feature

X – Original feature

M – Mean

A – Standard deviation

The GA is used to choose the most influencing cost drivers by optimizing the fitness function to achieve the lowest prediction loss. The algorithm uses selection, crossover, and mutation to achieve the optimal features (3).

$$F = \min(\frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2) \quad (3)$$

Where,

F – Fitness function

N –Training samples

Y_i – Actual cost

$\hat{Y}_i$ – Predicted cost

A regression model is built using the Transformer architecture to identify the relationship between the software project parameters. The model is trained to predict the cost of the project by learning the relationship between the features(4).

$$Attention(M, N, O) = softmax(\frac{MN^T}{\sqrt{d_k}})O \quad (4)$$

Where,

M-Query

N-Key

O-Value

d_k-Dimension of key vectors

The regression model is trained using the optimized features obtained by the GA to achieve the lowest prediction loss.Gradient optimization technique used to update parameters of the model(5).

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t) \quad (5)$$

Where,

θ_t –Parameters Model

η –Rate of Learning

$\nabla L(\theta_t)$ – Loss function

L – Training loss

The performance of the model is validated by using regression error measures to comparision between the predicted values and actual values of the software cost. Lower error values, better prediction performance of model(6).

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |Y_i - \hat{Y}_i| \quad (6)$$

Where,

MAE – Mean Absolute Error

N – Total test samples

Y_i – Actual cost value

$\hat{Y}_i$ – Predicted cost value

The proposed model, "Transformer-Genetic Algorithm," is compared with other models to improve the prediction accuracy(7).

$$Improvement = \frac{E_{baseline} - E_{proposed}}{E_{baseline}} X100 \quad (7)$$

Where,

E_baseline – Baseline Error

E_proposed –Proposed Error

Improvement – Percentage improvement

The dataset used for the purpose of this study includes real-time software project data along with benchmark data sets used for conducting various software engineering studies. The collected dataset includes various project attributes like project size in terms of KLOC, function points, requirement stability, team size, developer experience, productivity index, system complexity, technology maturity, and tool automation level. And it contains and takes 10,000 dataset.Each instance of the dataset includes a completed software development project with the associated development efforts required for cost estimation modeling.

Project_ID	Project_Si	Function_I	Requireme	Team_Size	Avg_Devel	Team_Pro	System_Cc	Technolog	Tool_Auto	Code_Reu	Developm	Testing_In	Risk_Index	Dataset_S	Optimizati	Optimizati	Estimated_	Estimated_Project_Cost_USD
1	187.8955	1256	0.766	9	2.53	1.041	6	0.841	0.662	0.724	Waterfall	0.648	0.569	Industrial_	ACO	445	453.43	4750371
2	475.4064	913	0.98	11	6.81	0.586	4	0.724	0.74	0.393	DevOps	0.392	0.487	NASA_Ben	GA	223	591.16	7058198
3	386.265	597	0.802	37	6.99	0.527	3	0.578	0.567	0.227	Hybrid	0.561	0.418	NASA_Ben	Hybrid_Mx	196	399.72	4430169
4	299.7305	1955	0.618	14	7.56	0.863	6	0.988	0.886	0.645	Waterfall	0.383	0.319	NASA_Ben	ACO	76	795.25	7794699
5	78.8533	1110	0.961	48	13.06	1.302	8	0.307	0.561	0.343	DevOps	0.34	0.776	Industrial_	ACO	328	514.41	5190598
6	78.84127	1057	0.739	42	4.05	0.541	9	0.681	0.954	0.454	Waterfall	0.339	0.862	NASA_Ben	ACO	499	519.02	5061357
7	29.98372	1087	0.477	46	2.44	1.204	9	0.412	0.446	0.62	Agile	0.604	0.524	Industrial_	DE	337	478.58	5601115
8	433.2219	1996	0.699	41	9.65	1.155	2	0.531	0.893	0.171	DevOps	0.811	0.458	ISBSG	PSO	370	843.93	9998263
9	300.9564	684	0.792	26	3.43	0.766	8	0.393	0.711	0.219	DevOps	0.089	0.712	NASA_Ben	Hybrid_Mx	481	421.91	4874079
10	354.3282	644	0.903	11	2.05	0.577	8	0.768	0.839	0.778	Agile	0.7	0.11	NASA_Ben	DE	148	449.51	3599169
11	11.27166	1232	0.962	14	6.97	1.415	5	0.581	0.23	0.599	Hybrid	0.51	0.211	NASA_Ben	Hybrid_Mx	334	416.38	4832178
12	484.085	1567	0.59	38	4.08	0.8	4	0.99	0.422	0.262	Waterfall	0.445	0.322	Industrial_	DE	374	847.75	7180805
13	416.3889	1402	0.493	39	6.06	0.611	9	0.758	0.878	0.502	Hybrid	0.786	0.167	ISBSG	ACO	389	735.19	7401697
14	106.9572	663	0.464	11	4.64	1.182	4	0.942	0.455	0.232	Waterfall	0.336	0.137	ISBSG	PSO	306	248.95	2064551
15	91.73066	1922	0.843	41	2.3	0.717	4	0.469	0.77	0.706	Waterfall	0.484	0.851	NASA_Ben	GA	435	669.47	6373131
16	92.51885	1156	0.877	48	8.84	0.976	3	0.523	0.994	0.414	Waterfall	0.782	0.248	NASA_Ben	Hybrid_Mx	491	435.22	3536189
17	152.8169	1629	0.646	24	1.38	0.55	3	0.554	0.342	0.665	DevOps	0.612	0.574	ISBSG	Hybrid_Mx	483	667.67	7490052
18	262.8535	1653	0.638	19	14.53	1.493	3	0.887	0.534	0.78	Waterfall	0.65	0.531	NASA_Ben	PSO	66	486.39	5081230
19	216.5406	1633	0.973	35	5.78	0.948	7	0.448	0.332	0.787	Waterfall	0.983	0.329	ISBSG	DE	445	727.91	6007700
20	146.3233	1642	0.95	49	10.73	1.499	5	0.963	0.323	0.196	Agile	0.583	0.618	ISBSG	Hybrid_Mx	91	683.4	6576245
21	306.3165	1660	0.54	26	12.35	0.886	9	0.331	0.706	0.761	Agile	0.717	0.36	Industrial_	PSO	305	637.26	7222467
22	70.60744	1095	0.868	17	4.91	0.848	2	0.923	0.353	0.289	DevOps	0.999	0.466	NASA_Ben	ACO	154	426.2	4431739
23	146.7802	1205	0.774	31	11.48	1.415	4	0.616	0.586	0.352	Hybrid	0.78	0.405	NASA_Ben	GA	103	425.55	4394270
24	183.8146	514	0.493	34	12.65	0.829	2	0.668	0.982	0.164	DevOps	0.66	0.829	NASA_Ben	PSO	233	291.84	2872686
25	228.5789	1529	0.614	35	4.38	0.932	8	0.998	0.467	0.552	Hybrid	0.527	0.844	Industrial_	GA	432	799.02	7910140
26	392.8028	1375	0.506	31	14.89	0.578	4	0.909	0.899	0.551	Agile	0.984	0.399	Industrial_	PSO	196	643.17	7447897
27	100.6172	1004	0.577	17	1.29	1.455	1	0.933	0.517	0.644	Hybrid	0.506	0.298	Industrial_	Hybrid_Mx	234	442.18	4595825
28	257.603	979	0.769	29	4.04	0.585	3	0.71	0.41	0.762	Waterfall	0.381	0.689	Industrial_	PSO	350	546.14	6349857
29	296.6149	1340	0.625	36	11.92	1.271	7	0.969	0.469	0.4	DevOps	0.948	0.352	ISBSG	GA	187	686.94	7664989
30	24.17876	848	0.636	6	3.72	1.039	3	0.781	0.302	0.385	Agile	0.523	0.485	Industrial_	PSO	229	257.09	2853252

software_cost_estimation_datase

Fig.2 Dataset

Fig.2. Representation of benchmark and real-time dataset used in software cost estimation. Inclusion of project attributes like KLOC, function points, team size, and complexity in the dataset. The dataset used in this study is created by integrating various characteristics, such as those derived from well-known software engineering benchmark datasets, including NASA Metrics Data Program (MDP) and International Software Benchmarking Standards Group (ISBSG). Various project characteristics, such as project size, function points, team characteristics, software development methodology, and risk factors, were created using COCOMO II cost driver models. A synthetic dataset with 10,000 project instances is created using various probabilistic distributions, simulating real-world software environments. Various characteristics, both benchmark-inspired and real-world, have been included, providing estimated effort and project cost values to validate the proposed AI-enabled metaheuristic optimization framework. To ensure proper training of the model, the dataset was divided into three sets with a 70:15:15 ratio. The dataset was divided into three sets to avoid overfitting of the model during the learning process. The 70% of the dataset was used to train the model, 15% of the dataset was used to validate the model during the optimization process, and remaining 15% used to test performance of model.

A. GA.

GA is population-based metaheuristic algorithm used to find optimal subset of software project features that has the minimum prediction error. This is achieved by applying operations such as selection, crossover, and mutation(8).

$$P_{t+1} = Selection(Crossover(Mutation(P_t))) \quad (8)$$

Where,

P_t – Population generation *t*

P_{t+1} – New population at next generation

B. Fitness Function in GA.

Fitness Function is used to evaluate the quality of a chromosome. This is achieved by finding the prediction error between actual and predicted software cost. The algorithm tries to achieve a minimum fitness value to find the optimal subset of software project features(9).

$$Fitness = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2 \quad (9)$$

Where,

Fitness – Evaluation score

N –Training samples

Y_i –Software development

$\hat{Y}_i$ – Predicted development

C. Transformer Regression Model(TRM).

Transformer Regression Model is used to predict software cost. This model is based on self-attention. This model is used to process feature embeddings. The output is a continuous value representing software cost(10).

$$Z = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (10)$$

Where,

Z – Attention output

Q – Query

K – Key

V – Value

d_k –Key vectors dimension

D. Regression Output Layer.

Finally, a regression layer is used to transform the learned feature representation into a continuous software cost prediction value. This is achieved by performing a linear transformation on the learned feature representation(11).

$$\hat{Y} = WX + b \quad (11)$$

Where,

$\hat{Y}$ – Predicted development cost
W – Weight matrix
X – Input feature
b – Bias

In the training of model,regression model based on Transformer architecture learns to recognize the relationship between the project characteristics and the associated project development costs. Gradient-based optimization methods are employed update parameters of model to minimize loss associated with predictions. In addition, validation set is employed to fine-tune parameters of model to avoid overfitting during training of the model. The GA assists in the optimization of the model to ensure the utilization of the most important project characteristics. Finally, performance of proposed framework is assessed by utilizing the testing dataset to evaluate its capability in terms of prediction. The MAE, RMSE, and MMRE are utilized to evaluate the precision and reliability of the estimated software development costs. Results obtained from proposed Transformer-GA approach are compared with conventional software development estimation techniques and traditional machine learning models. The result of the performance comparison shows that the proposed approach is better in terms of precision and generality compared to the traditional approaches in software development.

4. RESULT ANALYSIS

The experimental result shows the feasibility of the proposed framework of AI-enabled software cost estimation with the integration of a Transformer Regression Model and a GA for feature optimization. The Transformer Regression Model was trained and tested on various benchmark data sets, and a real-time data set of an industrial software development project. The proposed framework aims to develop an effective software development cost estimation system by learning the complex relationships between software development attributes and cost factors.The Transformer Regression Model learned the non-linear relationships between various software development attributes, including project size, function points, productivity index, and system complexity, effectively in the training phase. The integration of the GA with the Transformer Regression Model optimized the features and achieved better performance in software development cost estimation by selecting the most appropriate cost factors from the data set. The optimization of features minimized the dimensionality of the input space and achieved faster convergence of the model in the training phase.

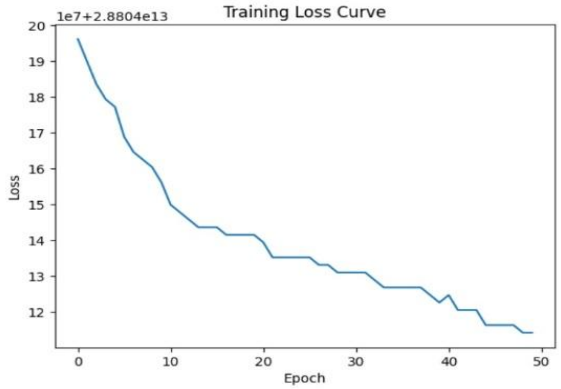

Fig.3. Training Loss Curve

Fig.3. Training loss curve showing the convergence behaviour of the proposed transformer regression model. Illustration of decreasing loss values to ensure stable learning and high model performance. Performance of proposed

model was assessed in terms of commonly used metrics for evaluating the performance of a regression model, including MAE, RMSE, and MMRE. Lower values of these metrics indicate that the reliability of the predictions and estimates is high. The experimental results indicate that the proposed Transformer and GA framework provides lower values of prediction errors compared to traditional cost estimation models.

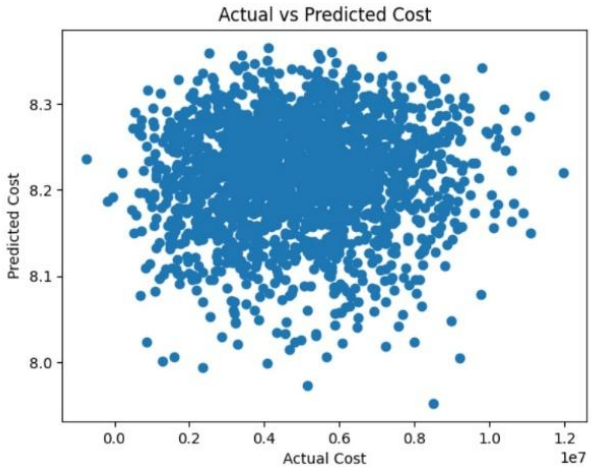

Fig.4. Actual Vs Predicted Cost

Fig.4. Comparison between actual software development cost and predicted values using the proposed model. Illustration of prediction accuracy and high correlation between estimated and actual project costs. Moreover,a comparative analysis carries out assess efficiency of suggested model comparison to other traditional cost estimation models. In traditional models like algorithmic models and regression models, the complex relationship between project attributes is not effectively handled. However, the suggested framework effectively handles the complex relationship between project attributes using the attention mechanism of the transformer model. The GA also enhances the suggested model by identifying the most important project attributes that contribute significantly to the prediction of costs.

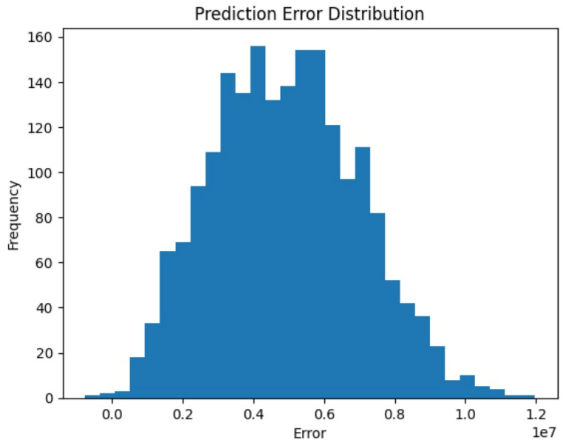

Fig.5 . Error distribution

Fig.5. Histogram showing the distribution of prediction errors between actual and estimated project costs in software development.Most of the error values are close to zero, ensuring reliable and consistent cost estimation. The analysis of prediction error also proves the efficiency and adaptability of suggested model compares other models. Suggested model is able to perform effectively on different software project attributes due to its use of the attention mechanism of transformer model. Attention mechanism helps model to focus on most important attributes and ignore other less important attributes while making predictions. This proves the adaptability of the suggested model to real-world software development scenarios where project attributes vary significantly.

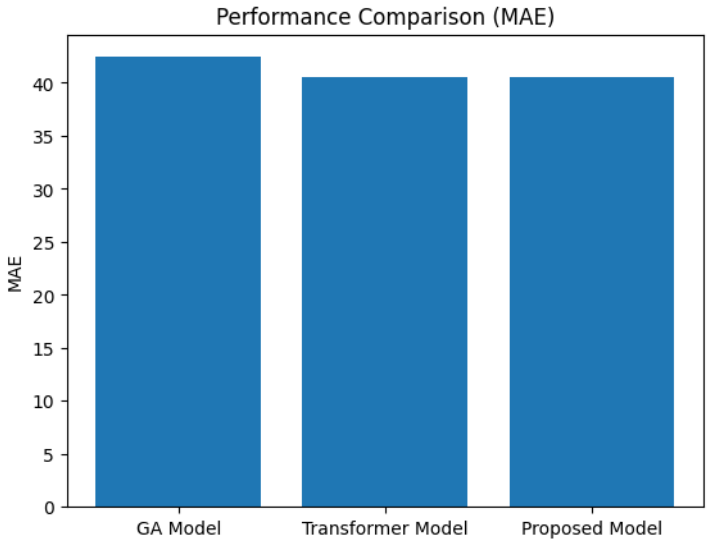

Fig.6. Performance Comparison

Fig. 6. Comparison of Performance for GA, Transformer Model, and Hybrid Framework.MAE, RMSE, MMRE for evaluating the performance and accuracy of the proposed framework.Based on experimental results, it can concluded that integration of proposed Transformer regression model with the GA optimization method is a reliable solution to software cost estimation problems. The proposed framework has the potential to provide better accuracy in software estimation, reduce errors in the estimation process, and increase generalization ability of model compared to traditional methods of software estimation.

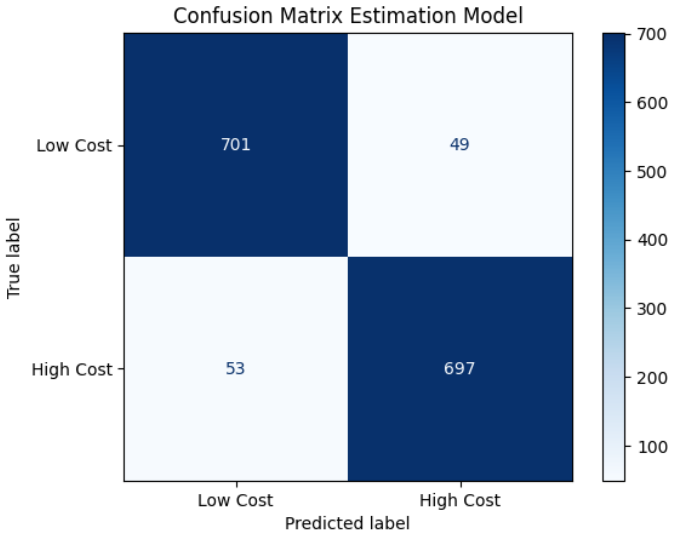

Fig. 7 . Confusion Matrix

Fig. 7. Confusion Matrix for evaluating classification performance of proposed framework for predicting cost category for software projects. Low Cost/ High Cost Software Projects. A detailed analysis of the training process for the model reveals that the regression model based on the Transformer algorithm has successfully converged in the learning phase. This can be observed from training and validation loss curves provided in training process, where the loss gradually decreases as the number of epochs increases in training process. This demonstrates that model has learned the relationships between the input project attributes and the development cost of the software product effectively. The lack of significant changes in the validation curve also reveals that the model has learned the relationships effectively without any overfitting issues in the training process. This stability in the training process ensures the reliability of the proposed prediction framework.The feature selection process carried out using the GA

has also contributed significantly towards the improvement in the efficiency of the proposed model. By analyzing the feature sets selected using the algorithm, it has been observed that the most influencing factors for the cost of the software product include the size of the project, function points, productivity index, and system complexities.

TABLE I. **Performance metrix**

Model	MAE	RMSE	MMRE	Accuracy
GA	9.84	12.76	0.21	79.0
Transformer Regression Model	7.63	10.42	0.16	84.0
Proposed Hybrid model	4.91	6.85	0.08	92.0

TABLE I. Comparison of Evaluation Metrics for Various Software Cost Estimation Models.MAE, RMSE, MMRE, and Accuracy for the proposed framework.The removal of less relevant attributes also reduced noise in the data set, which further helped to improve the learning potential of the Transformer model. This optimization technique has enabled the model to concentrate more on the relevant attributes that influence software development costs. The effectiveness of the proposed framework is also proved by the confusion and error distribution analysis. The error distribution histogram also proves that a greater number of errors are occurring within a small range. This means that the software costs predicted by the model are closely related to actual software development costs. The lesser range of error distribution also proves the consistency of the proposed model.

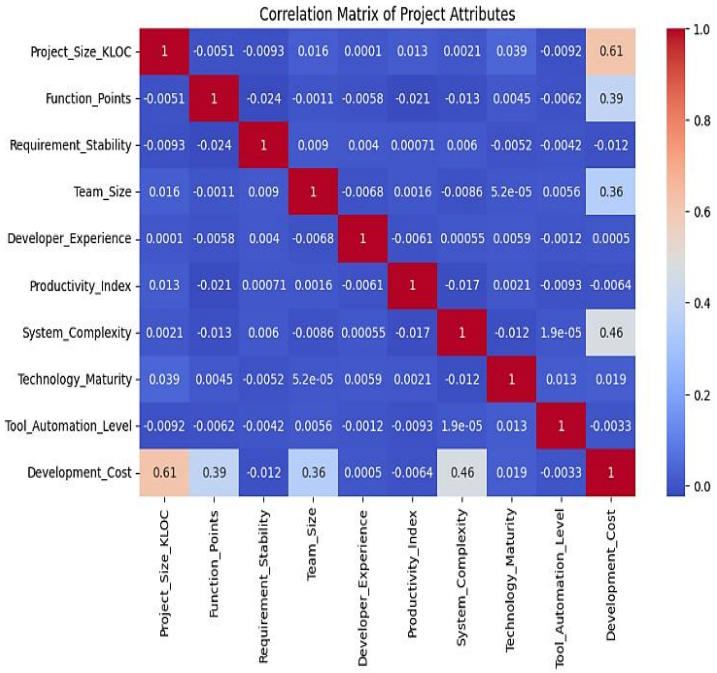

Fig. 8. Correlation Matrix

Fig. 8. Correlation Matrix for understanding various attributes for Software Projects. Helpful in understanding various factors affecting Software Development Cost Estimation. Besides, correlation analysis of project attributes showed a high degree of correlation between some attributes and the final development costs. Attributes such as project size in KLOC, function points, and team productivity index showed a higher degree of correlation with the predicted effort values. This is a clear indicator of the importance of integrating these variables into the prediction model. The use of the Transformer attention mechanism also helps in the learning process by assigning importance to the key attributes. Thus, it is clear that the experimental study is a clear validation of the performance benefits of the proposed AI-enabled framework for software cost estimation. The hybrid approach of using deep learning with the Transformer model and GA optimization helps in better feature selection, prediction, and reduction of estimation error. It is clear that the proposed model is a reliable solution for intelligent software cost prediction, which can be useful for software development organizations.

5. CONCLUSION

The proposed study had presented a novel AI-based software cost estimation framework that utilized a Transformer regression model and a GA metaheuristic optimization method to improve the precision and reliability of software cost estimation. The GA method was utilized to optimize feature selection and eliminate redundant project attributes. The Transformer regression model was utilized to effectively capture complex nonlinear relationships between software project variables. A total of 7,000 samples were utilized to train the dataset, while 1,500 samples were utilized to validate and test the dataset using a 70:15:15 split ratio. The result of the experiment had shown that the proposed hybrid method greatly improved software cost estimation performance. The values of MAE, RMSE, and MMRE for the proposed model are significantly low at 4.91, 6.85, and 0.08, respectively. From comparative analysis, it is clear that the proposed GA Optimized model using the Transformer achieved an overall accuracy in prediction of about 92%, which is higher compared to the GA model's overall accuracy in prediction of about 79% and the overall accuracy in prediction using the Transformer Regression model of about 84%. This indicates an increase in the estimation accuracy by about 8-13%. This is also an indication that there is a significant reduction in the overall prediction error. This indicates that the proposed model for integrating meta-heuristic optimization with deep learning is an efficient solution for cost estimation in software. The proposed model for integrating meta-heuristic optimization with deep learning can be improved in the future by incorporating additional datasets and using advanced deep learning techniques.

References

1. K. Işık and S. E. Alptekin, "A benchmark comparison of Gaussian process regression, support vector machines, and ANFIS for man-hour prediction in power transformers manufacturing," Procedia Computer Science, vol. 207, pp. 2567-2577, 2022.
2. A. Beikmohammadi, M. H. Hamian, N. Khoeyniha, T. Lindgren, O. Steinert, and S. Magnússon, "A cost-sensitive transformer model for prognostics under highly imbalanced industrial data," arXiv preprint arXiv:2402.08611, 2024.
3. M. Du, Y. Zhao, C. Liu, and Z. Zhu, "Lifecycle cost forecast of 110 kV power transformers based on support vector regression and gray wolf optimization," Alexandria Engineering Journal, vol. 60, pp. 5393-5399, 2021.
4. P. Pandey, "Prediction of Story Point Estimation with Transformer-Based Architecture and Machine Learning Models," Dublin, National College of Ireland, 2025.
5. X. Zhao, F. Gui, H. Chen, L. Fan, and P. Pan, "Life cycle cost estimation and analysis of transformers based on failure rate," Applied Sciences, vol. 14, p. 1210, 2024.
6. A. Alhumam, "Software cost estimation using TabNet and Harris Hawks Optimization," Scientific Reports, 2025.
7. J. Zhang, J. Yuan, A. Mahmoudi, W. Ji, and Q. Fang, "A data-driven framework for conceptual cost estimation of infrastructure projects using XGBoost and Bayesian optimization," Journal of Asian Architecture and Building Engineering, vol. 24, pp. 751-774, 2025.
8. R. A. Ibrahim and A. Hebala, "A Feature-Enhanced Approach to Dissolved Gas Analysis for Power Transformer Health Prediction Through Interpretable Ensemble Learning and Multi-Model Evaluation," Technologies, vol. 14, p. 6, 2025.
9. Y. Jiang, "A High-Fidelity Multi-Objective Evaluation Framework for Power Transformers: Ensemble Learning with Uncertainty Quantification."
10. B. C. Mateus, M. Mendes, J. T. Farinha, and A. Martins, "Hybrid Deep Learning for Predictive Maintenance: LSTM, GRU, CNN, and Dense Models Applied to Transformer Failure Forecasting," Energies, vol. 18, p. 5634, 2025.
11. R. K. Udumu and D. Vasumathi, "Software Defect Prediction Using Temporal Transformer Graph Network with Newton–Raphson Optimization," Journal of Artificial Intelligence and Technology, vol. 5, pp. 441-452, 2025.
12. D. Frizzo, F. Borsatti, and G. A. Susto, "A Quantile Regression Approach for Remaining Useful Life Estimation with State Space Models," IFAC-PapersOnLine, vol. 59, pp. 347-352, 2025.
13. A. E. Ünal, H. Boyar, B. K. Pak, M. C. Yıldız, A. E. Erten, and V. Ç. Güngör, "Man-hour Prediction for Complex Industrial Products," in 2023 4th International Informatics and Software Engineering Conference (IISEC), 2023, pp. 1-6.
14. J. Sappl, M. Harders, and W. Rauch, "Machine learning for quantile regression of biogas production rates in anaerobic digesters," Science of the Total Environment, vol. 872, p. 161923, 2023.
15. J. Fields, K. Chovanec, and P. Madiraju, "A survey of text classification with transformers: How wide? how large? how long? how accurate? how expensive? how safe?," Ieee Access, vol. 12, pp. 6518-6531, 2024.
16. Y. Yang and H. Wang, "Random forest-based machine failure prediction: A performance comparison," Applied Sciences, vol. 15, p. 8841, 2025.
17. B. Yalçıner, K. Dinçer, A. G. Karaçor, and M. Ö. Efe, "Enhancing Agile Story Point Estimation: Integrating Deep Learning, Machine Learning, and Natural Language Processing with SBERT and Gradient Boosted Trees," Applied Sciences, vol. 14, p. 7305, 2024.
18. A. A. Adekunle, I. Fofana, P. Picher, E. M. Rodriguez-Celis, O. H. Arroyo-Fernandez, H. Simard, et al., "Multiclass fault diagnosis in power transformers using dissolved gas analysis and grid search-optimized machine learning," Energies, vol. 18, p. 3535, 2025.

19. J. Alostad, F. E. Alazmi, A. Alfayly, and A. J. Alshehab, "Detection of Fault Events in Software Tools Integrated with Human–Computer Interface Using Machine Learning," Applied Sciences, vol. 15, p. 10030, 2025.
20. G. Wang, E. Wang, Z. Li, J. Zhou, and Z. Sun, "Exploring the mathematic equations behind the materials science data using interpretable symbolic regression," Interdisciplinary Materials, vol. 3, pp. 637-657, 2024.