Software cost estimation Using Machine Learning Methods

Diksha Sharma, 21BCE058
Computer Science and Engineering
Institute of Technology, Nirma University
Ahmedabad, India
21bce058@nirmauni.ac.in

Nupur Gandhi, 21BCE074
Computer Science and Engineering
Institute of Technology, Nirma University
Ahmedabad, India
21bce074@nirmauni.ac.in

***Abstract*—In the software development industry, accurate cost estimation is vital for effective project management. This paper explores the use of four machine learning models for software cost estimation: Linear Regression, Random Forest Regressor, Gradient Boosting Regression, and XGBoost. These models are crucial for precise project planning and resource allocation. The paper also delves into the complexities of data collection, preprocessing, and the selection of evaluation metrics. The dataset consists of features representing various attributes of an organization and its software development process from its product, process, size, users, and developers to the environment and the general information of the organization. The results on implementation showed that the XGBoost regression model works best with an r2 score of 0.902854. By analyzing these models, their attributes, and real-world scenarios where they excel, this paper contributes to enhanced software cost prediction with accurate predictions. In a world where software projects drive many industries, these models serve as guiding tools for project success.**

***Index Terms*—Software cost estimation, Data Pre-processing, One-Hot Encoder, Machine learning algorithms, Linear regression, Random forest regression, Gradient Boost, XGBoost, Grid Search**

I. Introduction

Software cost estimation holds a pivotal role within the realm of project management in the dynamic software development industry. The ability to derive precise and dependable estimates is paramount for orchestrating projects efficiently. [1] This necessitates a profound understanding of the intricacies involved in software project management as it guides resource allocation and overall project planning. It hinges on precise parameter assessment, including cost, development time, effort, risk analysis, and, notably, team size estimation. [2] To ensure consistency and efficiency in project development, these estimations must be conducted early on, relying on dependable models.

However, the software industry often struggles with unreliable estimates, and no single effort estimation model has consistently excelled in predicting software project effort under all circumstances. As a result, software engineering researchers have been continuously introducing new models to enhance the accuracy of effort prediction. In a systematic review, Jørgensen and Shepperd [3] identified 11 estimation approaches across 304 selected journal papers and they fall into two major categories: non-algorithmic and algorithmic models.

Algorithmic models, which often depend on mathematical formulas and project size estimates, have been widely utilized in software projects. They use analytical or statistical equations to link software project costs with various project features. Examples include COCOMO and the SLIM Model. However, these models frequently yield varying estimates for software development effort and cost, posing challenges for cost and resource management.

Non-algorithmic models, including machine learning-based approaches like Artificial Neural Networks, Fuzzy Logic, and Evolutionary Computation, have gained prominence. They offer adaptability and are effective in real-world scenarios, identifying correlations between input cost parameters and output cost development effort. Unlike conventional models, they address complex relationships between contributing factors and have the capability to handle diverse data types making minimal assumptions about the relationship's form. [4] Machine learning-based techniques are instrumental in establishing correlations between critical input parameters and cost development efforts.

This paper embarks on a comprehensive exploration of four machine learning models meticulously employed for the purpose of software cost estimation: Linear Regression, Random Forest regression, Gradient Boosting Regression, and XGBoost. These models, characterized by their distinctive approaches, harbor the potential to provide valuable insights into the intricate landscape of software cost prediction. As this paper delves into the intricacies of each model, their inner workings, their capabilities, and the scenarios in which they shine the brightest are minutely scrutinized.

In addition, this paper delves even deeper into the science of software cost estimation. Beyond the models themselves, the entire process, from data collection to preprocessing is performed. Further, the complexities of the data landscape are deciphered, ensuring it is well-prepared for the subsequent predictive modeling. An essential component of this journey is the meticulous selection of evaluation metrics. Through an array of key metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and the R2 score, the performance of these machine learning models is assessed and analyzed.

In a world where software projects are the lifeblood of many industries, these models pave the way for more informed decisions, meticulous resource allocation, and precise cost control. They are the guiding compass for project managers and stakeholders in their quest for software development success.

This research paper is structured as mentioned. Section II contains a detailed analysis of existing literature. Section III contains a description of all the models implemented in this paper. Section IV presents the workflow of the article. Section V encompasses the results and discussion of the implemented models, followed by the conclusion and future work in section VI.

II. Literature review

Reliable and accurate software effort estimation is a critical success factor throughout the software development life cycle, profoundly influencing project feasibility. The field of software effort estimation has witnessed numerous methods and research studies proposing various models to predict and enhance software effort estimation. Some of them are discussed as below:

Mustafa et al. [5] used four machine learning algorithms, namely Artificial Neural Network (ANN), Support Vector Machines (SVM), K-star, and Linear Regression, to estimate software effort based on project features. The study employs a public dataset called Usp05-tf, which contains data from 76 university students' software projects. The results indicate that SVM provides the most accurate predictions with the lowest MAE value(approximately 2.6), while linear regression exhibits comparatively poorer performance.

Mahmood et al. [6] conducted experiments using Weka software to compare the performance of 13 machine learning methods on two datasets. Results indicated that the selected attributes, when compared to the full dataset, generally led to better model performance, with improvements in metrics such as R^2, MAE, RMSE, RAE, and RRSE. Linear Regression and SMOreg often performed well. However, the choice of evaluator, algorithm, and search method influenced the selected attributes' quality. Additionally, reducing the number of attributes typically improved model performance.

Bilge et al [7], in this study, assessed machine learning methods for software effort estimation using three datasets and varied training set sizes. Results consistently showed that reducing the training set size increased the mean magnitude of relative error (MMRE), highlighting the need for larger training datasets. Methods like MLP, RBF, and SVR improved with Principal Component Analysis (PCA) for dimension reduction. When compared to the COCOMO parametric model, these machine learning methods outperformed, particularly in datasets mirroring modern software development practices.

In a study conducted by Pandey [8], a comprehensive analysis was performed on various techniques and approaches employed in software cost estimation. The aim was to highlight the key benefits and limitations associated with each method. The study's conclusion emphasized the significance of considering all project factors, as their importance and impact can vary between projects. Qualitative factors such as team experience, development environment, and organizational culture, along with quantitative factors like project size and resource availability, were identified as essential components of estimation metrics.

According to Wen et al.'s [9] extensive analysis of 143 publications related to effort estimation, their findings align with the notion that Support Vector Machines (SVM) outperform other algorithms, exhibiting an average Mean Magnitude of Relative Error (MMRE) of 0.34 and a PRED(0.25) accuracy of 72%. In contrast, alternative machine learning algorithms like Artificial Neural Networks (ANN) displayed lower accuracy (MMRE = 0.37, PRED(0.25) = 64%). It's worth noting that many of these models were constructed using relatively small and homogeneous legacy datasets, such as COCOMO, Desharnais, or Albrecht, which may result in issues of underfitting or overfitting.

A different study adopted an ensemble approach using Artificial Neural Networks (ANN) and Linear Regression (LR), employing log transformations for both effort and duration estimation based on the ISBSG dataset. Their effort estimation results were comparable to the current study: MMRE = 0.22, PRED(0.25) = 81%, and MBRE = 0.29. However, their duration prediction model showed poor accuracy: MMRE = 0.88, PRED(0.25) = 27%, and MBRE = 1.

In a separate study by López-Martín and Abran [10] using the ISBSG dataset, ANN algorithms with log transformation of dependent variables and cross-validation were compared. The MLP model achieved remarkable accuracy, with MMRE = 0.18 and PRED(0.25) = 80%, even surpassing the MLP in the current study, though not as good as the top-performing algorithm, SVM.

Comparing the results of ensemble models developed by Pospieszny et al. [11] using the ISBSG dataset and GLM, MLP, and CHAID decision trees, MMREs were found to be 0.19 and 0.21. However, differences in PREDs (0.25) at 62% and 66% suggest that the data preprocessing approach in this study, incorporating SVM in the ensemble model and using cross-validation for evaluation, led to reduced error dispersion and more accurate estimates. The variance in prediction accuracy, even with similar databases and algorithms, underscores the importance of data quality and preparation methods.

III. Models Used

In this paper, four machine learning models are used and they are discussed below.

A. Linear Regression

Linear regression analysis is a supervised machine learning algorithm used to predict the value of dependent variables based on the value of the independent variables. It models the relationship between a response variable (dependent variable) and one or more predictor variables (independent variables) in a linear fashion. It aims to find the best-fitting line to the data.

The model is represented by an equation where 'y' is the response variable, 'x' is the predictor variable, 'm' is the slope, and 'b' is the intercept which can be expressed as a straight line equation as in equation 1.

$$y = mx + b \tag{1}$$

B. Random Forest Regressor

Random Forest Regression is a supervised learning algorithm that uses an ensemble learning method for regression. It is a commonly used bagging technique that combines the output of multiple decision trees to reach a single result. [?]

Here in random forest regressor all the decision trees run in parallel. Thus, there is no interaction between the trees while they are building and they are independent of each other. While constructing the decision trees all the features are not taken into consideration. Different trees are constructed on the basis of different combinations of various features of the dataset.

Feature space is reduced here since all the features are not considered. As individual decision trees are constructed in this model the final result is based on the mean of all the individual outputs.

The important parameters considered for increasing the prediction power of the model are the number of estimators which gives the number of decision trees, and max features which tells how many features to consider while splitting the node.

C. Gradient Boosting Regression

Gradient boosting Regression is an ensemble learning category model that combines the prediction results of several weak models to create a strong one. It builds models sequentially with each new model correcting the errors of the previous ones.

Gradient Boosting is known for its accuracy and prediction speed. The main objective of the gradient boosting model is to minimize the loss function. It is a robust model as it updates the weight of the previous model based on the gradient which are less prone to outliers.

D. XGBoost

XGBoost(Extreme Gradient Boosting) is a decision tree-based ensemble machine learning algorithm which is a more optimized version of gradient boosting and uses the bagging technique. It attempts to predict the value of the target variable by training multiple decision trees and combining all the intermediate results to obtain the final result.

It is well known model for its ability to handle large complex datasets and handling missing values. It incorporates regularisation techniques such as Lasso and Ridge regularisation, to prevent the overfitting of data and improve the model generalization. This produces a prediction model with better accuracy. This model is optimized and highly efficient in terms of speed as well as memory usage. [13]

IV. Workflow

The workflow used in this paper is described as shown in Figure 1. The process starts with the collection of data and before selecting the regression model for training and validation, data preprocessing makes a significant step along with its exploration and analysis. Then the cost of the software is predicted. After predicting the software effort, the regression models are evaluated and the stages from validation to evaluation are repeated again and again by hyper-tuning the parameters at each iteration until the best model with the highest accuracy is achieved.

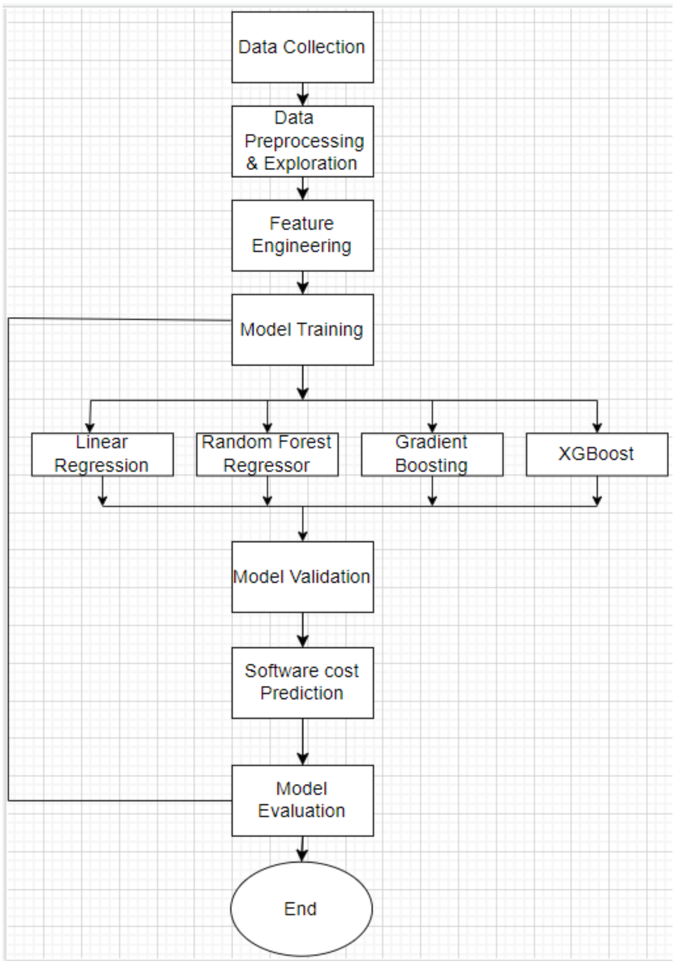

Fig. 1. Workflow used in the implementation

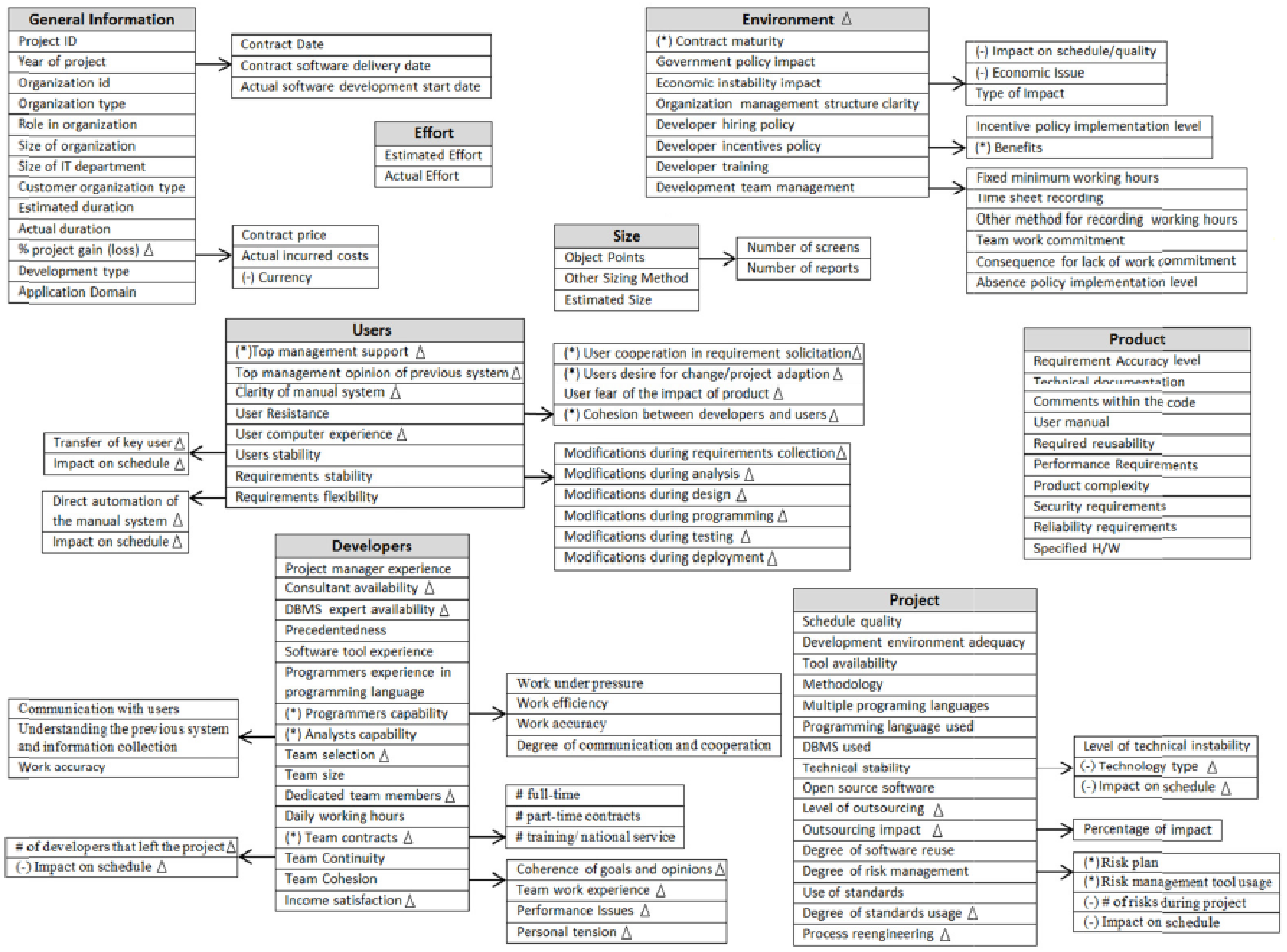

Fig. 2. Features used

A. Data collection and description

The research begins with the data collection. The dataset used is selected from Kaggle. [14] It consists of details of 120 software projects developed by various organizations around the world and 70 columns that represent various attributes of the organization and its software development process. The dataset is mainly divided into seven categories: general information, size, users, developers, environment, projects, and products. The features present in each category is shown in Figure 2.

B. Data Preprocessing

After the collection of data, data preprocessing is a necessary step before using any predictive algorithm. Data preprocessing is basically analyzing the dataset on how it is present, its variables, and its description. It is a crucial stage in ensuring the reliability and accuracy of predictive modeling, as the initial dataset directly affects the performance of the various prediction models. Thus although the used dataset contains substantial information on all the stages and aspects involved in software development; from the details of organizations, users, and developers to the environment and external factors affecting it such as governmental policies etc:-, it needs a considerable amount of refining and enhancement through various adjustments.

First, the dataset is analyzed descriptively. It contains the minimum and maximum values, the total count and frequency, the total number of unique values, the mean, standard deviation, and the three quartile values(25%, 50%, 75%) of each feature present in the used dataset. It is mainly computed to gain insights into the data's central tendencies and distribution. This step provides an understanding of the range and distribution of each feature. Next, it is checked for any missing or abstract values present in any of the features, and its treatment is carried out lest the dataset leads to a biased prediction model. It is found to contain many tuples with 'N/A' or '?' as its value. It is dealt with by first converting all the columns in the numeric format and then filling these abstract tuple values by the mean of the rest of

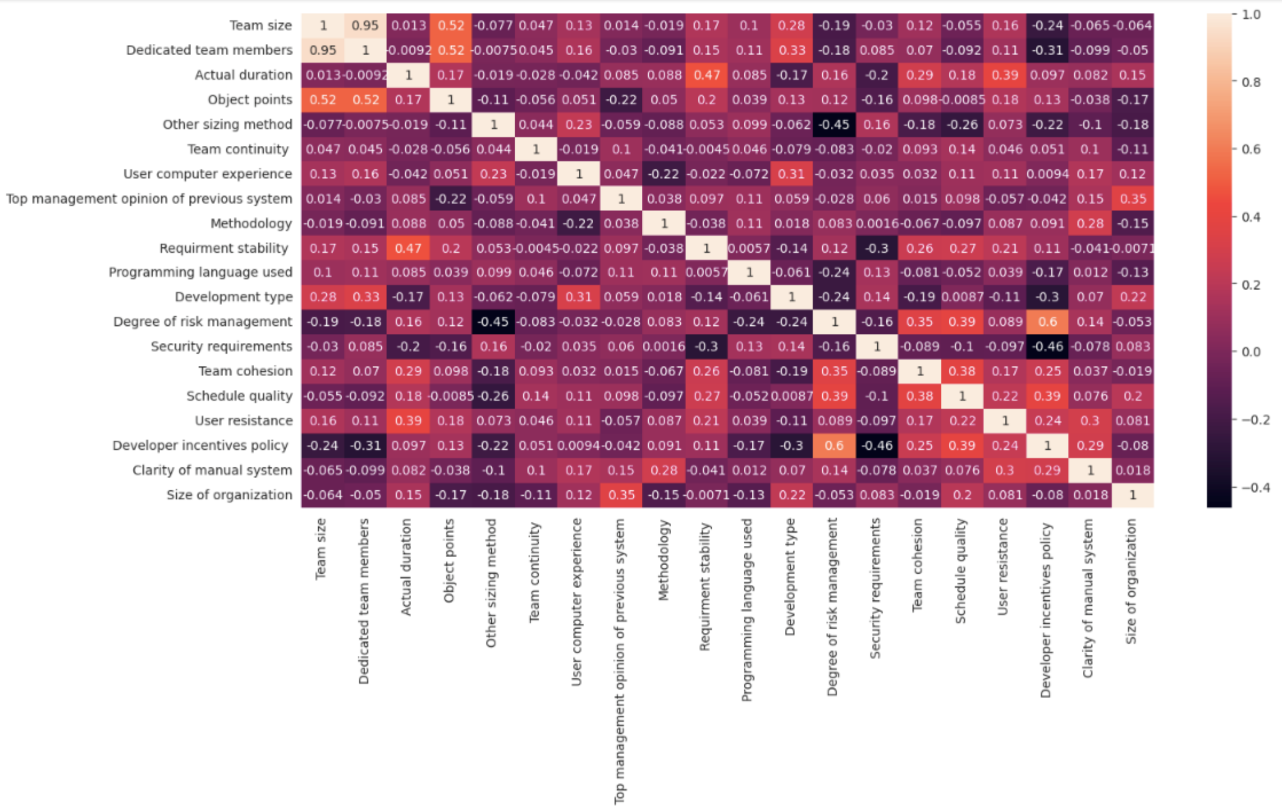

Fig. 3. Heatmap depicting the relationship between all the features

the values of that particular feature. Now the dataset becomes appropriate for further processing. [15]

Now the attributes 'Project ID' and 'Project Year' are dropped from the dataset as it is not relevant to the prediction models to be used in this paper. However, there still remain 68 features of which not all hold significance and will only serve complexity to the prediction model to be further used in this paper. A feature selection technique is implemented whose goal is to find the best set of features to help build optimized and accurate models of studied phenomena. It simplifies models, improves speed, and prevents a series of unwanted issues arising from having so many features. Therefore, the feature importance score for all the variables present in the used dataset is calculated. It refers to a technique that assigns a score to all the input features based on their importance in predicting the output variable. This score is calculated using *'ExtraRegressor'* model and plotted on a horizontal bar graph. It shows the score of each attribute and the most relevant features for the prediction model is found to be 'Team size', 'Dedicated team members', 'Methodology', 'Programming languages used' and etc:-. The top 20 most important features are selected for the implementation of the machine learning algorithms in this paper and the rest are discarded.

Then, a heatmap is generated as shown in Fig 3 to visualize the correlation between all the features selected. A heatmap basically provides a comprehensive view of feature relationships as it uses a system of color coding to represent different values. It is very useful in visualizing the concentration of values between two dimensions of a matrix helps in finding patterns and gives a perspective of depth. Now the pre-processed dataset containing only 20 features also holds some categorical attributes that are beyond the used prediction model's intelligence so for it to be interpretable by the machine, it is converted into numerical format using '*OneHot Encoder*'. One-hot Encoding technique is used for nominal features and for every categorical feature, it creates a new dummy variable which is further mapped with binary values containing either 0 or 1. Here, 1 represents the presence and 0 represents the absence of that value. [16] Thus, the new dataset containing 95 records and 20 attributes is now ready for further model development.

C. *Model development and implementation*

In the above section, the original dataset was tweaked and engineered and is now ready for model development. For model development i.e. making the dataset ready to be used in model, the following three steps are performed:

- **Declaration of dependent and independent variables**:- as a dependent variable as this is the aim of this paper i.e., to predict the cost incurred in the software development. The rest of the features present in the dataset after data pre-processing in their modified form are considered as the input variables.

- **Data splitting:-** The dataset is split into two parts 'train dataset' and 'test dataset' in the ratio of 4:1 on a random state of 365. The Train dataset is used to fit the model and then the test dataset is used for its evaluation to check how well it can generalize to the new and unseen data i.e. test data.

For each prediction model, the Grid search technique is used for hyperparameter tuning. It is the process of adjusting the parameters within a model to improve its performance on the new, unseen data. Grid search systematically explores the different combinations of these parameters and selects the best one suitable for the given model. The parameters selected for each model along with their values after its hyper-tuning are listed below:

TABLE I
THE PARAMETER VALUES AFTER HYPER-TUNING FOR LINEAR REGRESSION.

Parameters	Values
max. iterations	100, 200, 300,500, 1000
learning rate	0.1, 0.2, 0.001, 0.0001

1) Linear Regression: Here, max. iteration is a hyperparameter that can be set explicitly. It specifies the maximum number of times the algorithm will iterate on the training data and alpha is the learning rate. It is used to get the best descent in the curve. Too small values of alpha will make our model complex and will overfit the data while using too large values we may not be able to see the convergence in the graph as the model will be too simple and underfiting may occur.

TABLE II
THE PARAMETER VALUES FOR HYPERTUNING IN RANDOM FOREST REGRESSION

Parameters	Values
no. of estimators	100, 200, 300, 500, 1000 , 1500
random state	42

2) Random Forest Regressor: Here in random forest regressor no. of estimators denotes the number of decision trees made to train the data. n jobs is the hyperparameter which denotes the number of CPU cores or parallel jobs to be used. When set to -1 software automatically detects the number of cores available and uses them for parallel jobs.

3) Gradient Boosting: Max depth is the hyperparameter which denotes the maximum depth for each decision tree. min sample split is the hyperparameter which denotes the minimum number of samples required to split an internal node of each tree within an ensemble. min sample leaf is the hyperparameter which denotes the minimum number of samples required to be present in a leaf node of each tree within an ensemble.

4) XGBoost: Here, the subsample denotes the fraction of training data used to grow each decision tree during boosting.

TABLE III
THE PARAMETER VALUES FOR HYPER-TUNING IN GRADIENT BOOST REGRESSION

Parameters	Values
no. of estimators	100, 200, 300, 500, 1000
learning rate	0.01, 0.1, 0.2, 0.05
max depth	3, 4, 5, 6, 7, 8
min sample split	2, 5, 10
min sample leaf	1, 2, 4

TABLE IV
THE PARAMETER VALUES FOR HYPER-TUNING IN XGBOOST REGRESSION

Parameters	Values
no. of estimators	100, 200, 300, 500, 1000
max depth	3, 4, 5
learning rate	0.01, 0.1, 0.2
subsample	0.8, 0.9, 1.0
colsample bytree	0.8, 0.9, 1.0

Setting its value to 1 means using all the training data but it can lead to overfitting. When building trees in the sample, the no. of features(columns) to be randomly sampled is controlled by colsample_bytree.

D. Evaluation metrices

For each model, the following metrics were calculated:

- **Mean Absolute Error (MAE)**:- MAE is the accuracy measurement parameter which is used to calculate the absolute average difference between the predicted and actual values of the target variable. The formula used for its calculation is given in Equation 3.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2}$$

- **Mean Squared Error (MSE)**:- MSE is the accuracy measurement parameter which is used to calculate the average of the square of the difference between the predicted and actual values of the target variable as given in Equation 4.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3}$$

- **Root Mean Squared Error (RMSE)**:- RMSE is the accuracy measurement parameter that is used to calculate the square root of the average of the square of the difference between the predicted and actual values of the target variable. The formula used for its calculation is given in Equation 5

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4}$$

- **R2 Score**:- R2 score is the accuracy measurement parameter which gives values between 0 and 1 which shows how similar a regression line is to the data fitted. It helps in understanding the goodness of fit in the model. Its formula is given in Equation 6.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad (5)$$

The above-mentioned metrics tell a model's quality. The lower the MSE, MAE, and RMSE is, the better the model is however for a desirable model, a higher value of R2 score is required. These metrics help in assessing the performance of a predictive model and the best model is selected on specific characteristics of the dataset and the importance of error values.

V. Result and Analysis

After training the various models on training data and evaluating them on testing data as discussed in the previous section, Table V shows the accuracy results obtained.

TABLE V
Evaluation results for different iterations on each model

Model Used	R2 score
Linear Regression	0.8077
Random Forest Regression	0.8769
XGBoost	**0.9028**
Gradient Boost	0.8352

By comparing these results, it is observed that the R2 score for the XGBoost model is the best among all the than used model **R2 score which is 0.902854**. The R2 score is the measurement of the accuracy score. Higher the R2 score higher the model accuracy is. Its error percentage is less than the others as well.

XGBoost model has a significant advantage over others because of its versatility and ability to handle complex datasets. It gains an edge over the random forest model as it uses a sequential approach for the optimization of decision trees and reduces error with each iteration on the previous decision tree. It outperforms the gradient boosting algorithm as it incorporates regularisation(Lasso and Ridge) techniques, which helps prevent any possible overfitting. Thus, it is more robust and less prone to noise fitting in the data. It has in-built capabilities of handling missing data and providing insights into feature importance. it is efficient and reduces computational costs by hardware optimization. [17]

Figure 4 is the error graph for the XGBoost algorithm, as it was the most accurate model obtained. It can be seen from the graph, that the error for each data point obtained is between the range of 0% to 40%, the maximum error percentage being 40 and the minimum 0 i.e. the value of prediction is almost the same as the actual value. Here, error is the difference between the predicted price and the actual price.

Figure 5 shows the actual price of the car in the dataset and the predicted prices as obtained by each model.

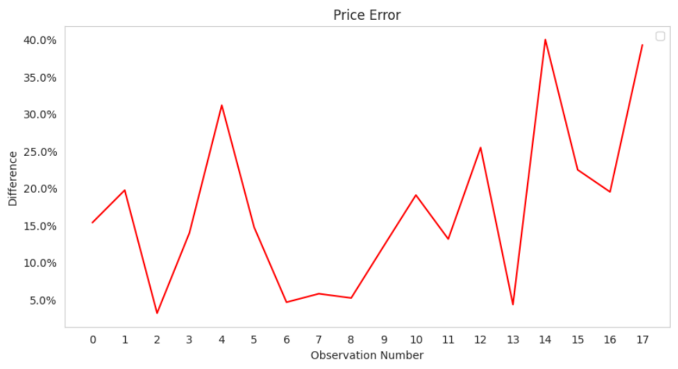

Fig. 4. Error percentage graph for XGBoost

VI. Conclusion

In conclusion, accurate software cost estimation is crucial for effective project management. This paper presents a comprehensive analysis of four machine learning models used for software cost estimation, namely Linear Regression, Random Forest regression, Gradient Boosting Regression, and XGBoost. Through rigorous evaluation, XGBoost emerges as the most accurate model, with a high R2 score and lower MSE, MAE, and RMSE values. The application of XGBoost, with its ability to handle complex datasets and incorporate regularization techniques, proves to be a valuable asset for software cost estimation. Accurate estimations enhance project planning, resource allocation, and cost control, ultimately contributing to the success of software development projects.

References

[1] Akhbardeh, F., & Reza, H. (2021, May). A Survey of Machine Learning Approach to Software Cost Estimation. In 2021 IEEE International Conference on Electro Information Technology (EIT) (pp. 405-408). IEEE.

[2] Idri, A., azzahra Amazal, F., & Abran, A. (2015). Analogy-based software development effort estimation: A systematic mapping and review. Information and Software Technology, 58, 206-230.

[3] Jorgensen, M., & Shepperd, M. (2006). A systematic review of software development cost estimation studies. IEEE Transactions on software engineering, 33(1), 33-53.

[4] Idri, A., Hosni, M., & Abran, A. (2016). Improved estimation of software development effort using classical and fuzzy analogy ensembles. Applied Soft Computing, 49, 990-1019.

[5] Hammad, M., & Alqaddoumi, A. (2018, November). Features-level software effort estimation using machine learning algorithms. In 2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT) (pp. 1-3). IEEE.

[6] Al Asheeri, M.,& Hammad, M. (2020, September). Improving software cost estimation process using feature selection technique. In 3rd Smart Cities Symposium (SCS 2020) (Vol. 2020, pp. 89-95). IET.

[7] Baskeles, B., Turhan, B., & Bener, A. Software effort estimation using machine learning methods. In 22nd international symposium on computer and information sciences (pp. 1-6). IEEE.

[8] P. Pandey, "Analysis of the Techniques for Software Cost Estimation," in 2013 Third International Conference on Advanced Computing and Communication Technologies (ACCT), Rohtak, India, 2013.

[9] Wen, J., Li, S., Lin, Z., Hu, Y., & Huang, C. (2012). Systematic literature review of machine learning based software development effort estimation models. Information and Software Technology, 54(1), 41-59.

[10] López-Martín, C., & Abran, A. (2015). Neural networks for predicting the duration of new software projects. Journal of Systems and Software, 101, 127-135.

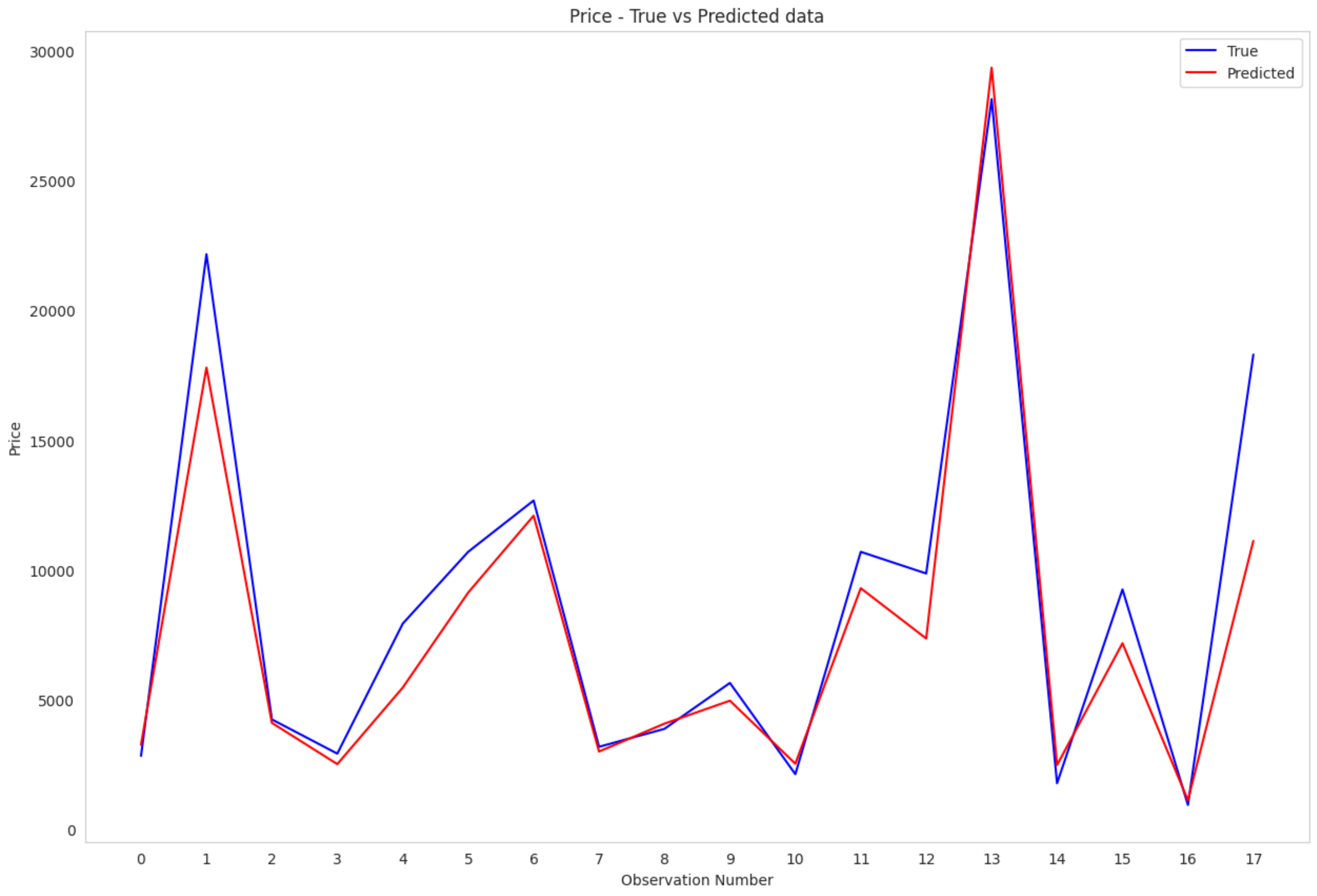

Fig. 5. The actual cost vs the predicted software cost

[11] Pospieszny, P., Czarnacka-Chrobot, B., & Kobyliński, A. (2015). Application of function points and data mining techniques for software estimation-a combined approach. In Software Measurement: 25th International Workshop on Software Measurement and 10th International Conference on Software Process and Product Measurement, IWSM-Mensura 2015, Kraków, Poland, October 5-7, 2015, Proceedings 25 (pp. 96-113). Springer International Publishing.

[12] Al Asheeri, M. M., & Hammad, M. (2019, September). Machine learning models for software cost estimation. In 2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT) (pp. 1-6). IEEE.

[13] Xia Z., Xue, S., Wu, L., Sun, J., Chen, Y., & Zhang, R. (2020). ForeXGBoost: passenger car sales prediction based on XGBoost. Distributed and Parallel Databases, 38, 713-738.

[14] Mustafa, E. I., & Osman, R. (2020, November). SEERA: a software cost estimation dataset for constrained environments. In Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering (pp. 61-70).

[15] Huang, J., Li, Y. F., & Xie, M. (2015). An empirical analysis of data preprocessing for machine learning-based software cost estimation. Information and software Technology, 67, 108-127.

[16] Krishnan, J. R., & Selvaraj, V. (2022, November). Predicting resale car prices using machine learning regression models with ensemble techniques. In AIP Conference Proceedings (Vol. 2516, No. 1). AIP Publishing.

[17] Chen, T., Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).