AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH (AIUB)


Report for the course of Software Project Management

# Software Effort Estimation Prediction Using Machine Learning Techniques

August 24, 2023

Prepared By:

MOMAHIDA-AN-NOOR (22-92685-3)
ABRAR AHMED KHAN (22-92681-3)
TAMANNA RAHMAN MALIHA (22-92711-3)
MD. PIAL HASAN (23-92900-1)

Supervised by:
Dr. S.M. Hasan Mahmud
Assistant Professor, Dept. of Computer Science
Ameriacan International University-Bangladesh (AIUB)

***Abstract*—Effort estimation is a critical task in software project management, and accurate predictions are essential for successful development. While machine learning algorithms have shown promise in effort estimation, their practical implementation remains limited. This paper aims to bridge the gap between research and practice by proposing a comprehensive approach that includes data preparation, model implementation, and maintenance. The study conducts a literature review, evaluating previous work on effort estimation and machine learning algorithms. It proposes a model that emphasizes data preprocessing and attribute selection to enhance estimation accuracy. The proposed model's performance is evaluated using the Google Colab. The paper acknowledges the challenges in effort estimation and highlights the importance of accurate estimation for project success. The research goals include analyzing previous work, studying machine learning schemes, proposing an enhanced model, and evaluating its performance. The related works section highlights previous research on machine learning-based effort estimation and the use of different algorithms, datasets, and attribute selection methods. The methodology introduces Support Vector Regression, Multilayer Perceptron, Linear Regression and Random Forest Regression algorithms and outlines their implementation steps. The paper concludes by proposing a practical and effective approach for implementing machine learning models in effort estimation, aiming to bridge the gap between research findings and practical deployment.**

***Index Terms*—Software project management, Effort estimation, Software development, Machine learning, Accuracy prediction, Data mining, Regression analysis, Evaluation metrics, ML algorithms, Testing, Resource allocation, Model prediction.**

## I. Introduction

Effort estimation plays a vital role in software engineering project management by predicting the necessary work units based on project features and insights from similar projects. Accurate estimation is critical for the success of software application development, as incorrect estimations can lead to project stagnation and failure. To address this challenge, various estimation methods, including expert estimation, algorithmic estimation, and machine learning, have been employed in practice. Among these methods, machine learning algorithms have gained prominence due to their ability to analyze historical data and provide accurate predictions. Despite the importance of accurate effort estimation, the practical implementation of machine learning techniques in software projects remains limited [1]. Existing research often focuses on finding the most accurate algorithm without considering implementation and maintenance aspects or addressing data preparation challenges. This gap between research findings and practical deployment hinders the widespread adoption of robust machine learning-based effort estimation models [2]. To bridge this gap, this paper proposes a comprehensive approach that encompasses data preparation, model implementation, and maintenance. The study begins with a literature survey, exploring previous works in effort estimation and classification parameters. It then conducts a thorough analysis of different machine learning schemes, evaluating their suitability and performance in effort estimation. Based on the findings, the paper proposes a model that emphasizes data pre-processing and attribute selection to enhance the accuracy and usability of effort estimation models. To validate the effectiveness of the proposed model, its performance is evaluated using the Google Colab tool [3]. The paper acknowledges the challenges faced by software teams and businesses in accurately estimating software effort, recognizing the significance of effort estimation for project success and risk reduction [2][3]. It highlights the different categories of effort estimation methods, such as algorithmic, non-algorithmic, and machine learning approaches. Moreover, it emphasizes the critical role of precise effort estimation in project planning, resource allocation, and decision-making processes. The primary objective of the study is to contribute to the field of effort estimation by evaluating and comparing various machine learning algorithms, providing insights into their suitability for different situations. By proposing a model that prioritizes data pre-processing and attribute selection, the paper aims to enhance the accuracy and usability of effort estimation models. The use of the Google Colab for performance evaluation adds further credibility to the study's findings.

The research goals of this paper are as follows:

(1) to perform an analysis and evaluation of previous work on effort estimation in software projects,

(2) to conduct a comprehensive study on various machine learning schemes and their classification,

(3) to propose a model that emphasizes data preprocessing and the selection of appropriate attributes to improve estimation accuracy, and

(4) to evaluate the performance of the proposed model using the Google Colab.

## II. Literature Review

Effort estimation in software development has been a subject of extensive research over the last two decades. Researchers have focused on using data mining, statistical, and machine learning algorithms to accurately forecast initial effort and duration during the early stages of a project [1][2]. Various algorithms, such as

artificial neural networks (ANN), case-based reasoning (CBR) models, and decision trees, have been tailored and compared for their performance in effort estimation [3].

Studies have shown that machine learning models generally outperform statistical models, with acceptable accuracy levels [2][4]. However, the choice of dataset and data preprocessing approaches can significantly impact the results, introducing disparities due to outliers, missing values, and the possibility of overfitting [5]. Different publications have presented diverse approaches and comparisons of algorithms. For example, artificial neural networks were compared with multiple regression models for effort estimation using COCOMO dataset, demonstrating the superiority of neural networks [6]. Effort estimation research has also explored the precision of different neural network types, using ISBSG dataset and employing normalization and cross-validation techniques [7].

In addition to algorithm selection, researchers have proposed ensemble methods, such as boosting, bagging, and random sampling, to improve prediction accuracy [8][9]. However, excessive use of ensemble methods can introduce performance overhead. Therefore, it is recommended to use a limited number of different algorithms and a simple ensemble approach, such as averaging estimates [8].

Data preprocessing plays a crucial role in training machine learning models, particularly in handling outliers and missing values, which can significantly impact accuracy [10]. Various techniques, such as deletion, imputation, and normalization, have been employed depending on the dataset. It is generally advisable to discard missing values, if possible, to avoid bias, and outliers are commonly removed using the three-standard-deviation rule [2][10].

Wen et al. (2012) conducted a comprehensive review of machine learning algorithms used for effort estimation, investigating 84 studies from the last two decades [11]. Their review highlighted the various machine learning techniques employed, such as decision trees, artificial neural networks, support vector machines, and Bayesian networks [11]. The study emphasized the importance of accurate effort estimation in software development projects.

Sharma and Vijayvargiya [12] explored the use of soft computing techniques, including linear regression, support vector regression, artificial neural networks, random forests, decision trees, and bagging methodology, to predict the time and resources required for software projects using benchmark datasets [12]. Their study demonstrated the applicability of these machine learning algorithms in improving effort estimation accuracy.

Other studies utilizing machine learning, such as k-nearest neighbor regression and support vector regression, further support the effectiveness of these techniques. Additionally, the paper emphasizes the importance of attribute selection to enhance model accuracy [11].

Various datasets, including NASA, COCOMO, and Desharnais, have been used for effort estimation, providing valuable resources for researchers [12]. The paper also recognizes the benefits of adopting agile methodologies, offering flexibility and improved accuracy through the modeling of programming patterns and misestimation patterns. Evaluation of effort and duration prediction models often utilizes measures such as mean magnitude relative error (MMRE) and percentage relative error deviation (PRED) [13]. While these measures have been subject to criticism, they remain widely used for comparison and are often supplemented with other measures like mean magnitude relative error to estimate (MMER), mean of balanced relative error (MBRE), and mean absolute residual error (MAR). Cross-validation procedures, such as k-fold, are commonly employed for validation to prevent overfitting.

Despite the extensive research conducted, the implementation of machine learning predictive algorithms for effort and duration estimation within organizations remains limited [2]. Challenges include outdated and small datasets, as well as the need for models to align with current software methodologies and development approaches [14]. This paper aims to address the limitations and recommendations proposed by researchers by outlining a practical and effective approach for building and deploying machine learning models for effort and duration estimation [2][8]. By considering the suggestions from previous studies, the goal is to enable the implementation of these models in practice.

## III. Methodology

Software effort estimation is anticipating the resources required to construct a software system, such as time, cost, and manpower. It is an important task in project management and influences decision-making processes. Organizations can enhance their accuracy in forecasting software work by employing machine learning algorithms such as SVR(Support Vector Regression), MLP(Multilayer Perception), XGBoost(Extreme Gradient Boosting), Linear

Regression and Random Forest Regression resulting in better project planning and resource allocation.

### *A. SVR (Support Vector Regression)*

SVR is a machine learning approach that is used for data analysis and regression problems. It is based on the Support Vector Machines (SVM) method, which was designed originally for classification problems. SVR seeks a regression function that maximizes the difference between data points and a decision boundary. A subset of the training data known as support vectors determine the decision boundary [2]. SVR is capable of handling both linear and nonlinear regression tasks. It achieves non-linearity by using kernel functions to transform the original feature space into a higher-dimensional space into which a linear regression model may be applied.

To use SVR first proceed by preparing the data for analysis. Handling missing values, scaling features, and encoding categorical variables are all possibilities [1]. Determine the features that will be analyzed. Choosing the correct characteristics for the SVR model boosts its performance and interpretability. Using labeled training data, train the SVR model. Before training the model, certain hyperparameters, such as the kernel type, regularization parameter (C), and kernel-specific parameters, must be set in SVR. Using approaches such as grid search or random search, you can determine the ideal combination of hyperparameters. Once you've trained your SVR model, you may use it to generate predictions on new, previously unknown data. The model will produce continuous values that represent the expected target variable.

The equation of SVR is given below,

$$\int(x) = \omega^T \varphi(x) + b$$

Here, $\omega$ is the weight vector and b is a bias term; $\phi$(x) is a function that maps the feature; x is the input vector.

### *B. MLP (Multilayer Perceptron)*

MLP is an abbreviation for Multilayer Perceptron, a form of artificial neural network used for data analysis and machine learning applications such as regression and classification. MLP is a feedforward neural network, which means that information goes from the input layer to the output layer in one direction via one or more hidden layers. First, Gather data for analysis. Create the MLP architecture. Set the number of layers, the number of neurons in each layer, and the activation functions. The input layer size should be proportional to the number of features in the data. Using labeled training data, train the MLP model. Forward propagation is used in the training process, where input data is transferred through the network and the output is determined [5]. The network's weights and biases are then updated via backward propagation to minimize the error between anticipated outputs and true labels. Evaluate the MLP model's performance on the testing data [2]. Depending on the job it will measure such as accuracy, precision, recall, and F1-score can be used for classification, as can metrics such as mean squared error (MSE). The number of hidden layers, the number of neurons in each layer, the learning rate, the activation functions, and regularization parameters are all hyperparameters in MLP. When a trained MLP model is obtained, it may be used to make predictions on new, previously unknown data. The model will output projected values or class probabilities depending on the job.

For the equation, the input features are assumed as X1, X2,. . . . . Xn.

Each hidden layer is made up of several neurons, and the output of each neuron is calculated by taking the weighted sum of the inputs and applying an activation function to it. The hidden layer is calculated as,

Oj = (W1X1+ W2X2+...+ WnXn+ Bj)

Here, W is considered as weights and B is bias; O is considered the hidden layer. Later on, the output layer produces the final prediction, which is

Output=(V1O1+V2O2+...+VmOm+Bv) in which, Vm are the weights that connect to the last hidden layer's neurons to the output neuron, Om is the output of the hidden layer and Bv is the bias term.

### *C. Linear Regression*

Linear regression is a basic supervised machine learning algorithm used to predict a target continuous variable based on one or more input characteristics [5]. This is a simple but powerful technique that forms the basis for more complex regression algorithms and is the building block for data analysis projects The primary goal of linear regression is to find the best fit linear relationship between the input characteristics and the target variable. The initial data generation involves the collection of data sets with predictable values and relevant input characteristics. Handling missing values, by insertion or deletion ensures data integrity. Scaling features can increase performance, especially for specific algorithms [2]. The transformation of categorical properties, typically through techniques such as one-hot encoding, supports machine-learning algorithms. This framework provides robust data, which is a reliable

foundation for accurate data analysis and predictive model development.

The equation of a simple linear regression model can be expressed as:

y=mx+b

Where, y is the predicted target variable, x is the independent variable, m is the slope of the regression line, which represents the change in y for a one-unit change in x, b is the y-intercept, which represents the value of y when x is 0.

In a more general form, for multiple linear regression with multiple independent variables, the equation is:

y = b0 + b1x1 +b2x2 + .....+bpxp

where, y is the dependent variable, x1, x2,...,xp are the independent variables, b0 is the y-intercept, b1, b2,...,bp are coefficients of the independent variables, representing how much each independent variable influences y.

The process starts by training a linear regression model using labeled data. The coefficients are generally generated through methods such as OLS or gradient descent, which reduce the predicted-true heterogeneity. The required metrics—MSE, RMSE, MAE, and R-squared—assess the consistency of the predictions with the actual data. Despite their simplicity, hyperparameters are important. Regularization (L1/L2) prevents overfitting; Feature selection Feature selection helps. After training and optimization, the model predicts new data using the known coefficients. This journey—from training to informed prediction—highlights its role in understanding data relationships and making valuable predictions. In this way, linear regression transforms into a powerful analytical and predictive instrument, bridging the gap between data insights and real-world applications.

### *D. Random Forest Regression*

Random forest regression is a powerful machine-learning technique for regression applications and data analysis [2]. It is based on the Random Forest algorithm, which was originally developed for classification. Like its classification counterpart, random regression forest works by constructing a variety of decision trees [3]. Each tree is trained on a random subset of the data and makes predictions based on the inputs.

The main objective of random forest regression is to develop a prediction model that minimizes the difference between the predicted actual values. This is arrived at by averaging the predictions of several different decision trees, providing more stable and accurate results.

The equation for a single decision tree in a Random Forest Regression is as follows:

Prediction = Bias + Σ (Tree Prediction)

Here,

• Bias represents a constant term that accounts for the average of all the target values in the training dataset.

• Tree Prediction is the prediction made by an individual decision tree in the Random Forest.

When random forest regression is used, the process begins by adding the data set to the factors related to the target variable. Data prep handles missing values and encodes hierarchical values to make them true. Training models consist of different decision trees, which are reduced into subsets with different features. Together, these trees predict the target, and the final result is the average of the predictions of the individual trees. Model evaluation uses standard metrics (MSE, RMSE, MAE) to measure performance. The clustering nature of random forests dominates overfitting, increasing generalization. Key hyperparameters—number of trees, depth, samples per leaf, selection of segments—require optimization. After training and optimization, the random forest model is ready for prediction. It uses a collection of tree views to provide reliable estimates of new information. This session on prep, ensemble modeling, and evaluation demonstrates the power of random forest regression to handle complex regression tasks and deliver more accurate forecasts.

### *E. MAE And RMSE*

Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are common metrics used to evaluate the performance of predictive models, particularly in regression tasks. Here's a brief explanation of each:

1. Mean Absolute Error (MAE):

• MAE is a simple metric that calculates the average of the absolute differences between the predicted values and the actual values.

• It gives equal weight to all errors, regardless of their magnitude.

• MAE is easy to understand and interpret. It represents the average magnitude of errors in the predictions.

• You want a straightforward measure of model performance.

• All errors, regardless of their direction (overestimation or underestimation), are equally important.

2. Root Mean Square Error (RMSE):

• RMSE is a more complex metric that calculates the square root of the average of the squared differences between the predicted values and the actual values.

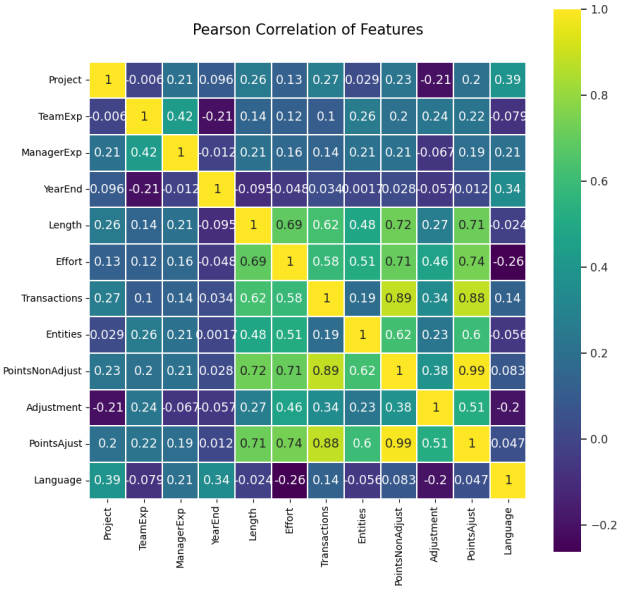


**Figure 1:** A dataset of Pearson Correlation of Features.

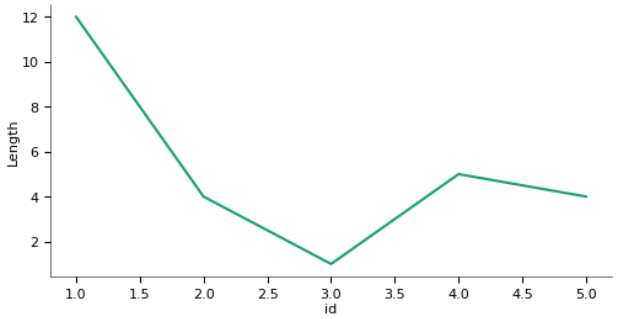


**Figure 2:** A dataset of Pearson Correlation of Features.

• It gives higher weight to larger errors, making it sensitive to outliers.

• RMSE is commonly used in situations where larger errors are more costly or problematic.

• You want to penalize larger errors more heavily.

• You want to measure the model's ability to predict accurately, considering both bias and variance.

In summary, we use MAE for a simple, easy-to-understand measure of model accuracy that treats all errors equally. RMSE, on the other hand, is preferred when you want to penalize larger errors more and take into account both bias and variance in your model's predictions. The choice between MAE and RMSE depends on the specific problem and the importance of different types of errors in your application.

### *F. Dataset*

The dataset that has been collected contains information about a multitude of software projects that have been previously completed. Within the dataset, the topics that have been gathered include project number, team experience, manager experience, year end, length, effort, points adjustments, adjustments, and language. Our primary objective with this dataset is to utilize the other columns to predict the effort estimation. In order to achieve a more effective and efficient dataset, we preprocessed the data to remove any extraneous information that had no bearing on the effort. We identified and removed any duplicate and null values. To gain a better understanding of the values we are working with, we provided an overview of the range of our dataset parameters. To achieve our primary objective of predicting effort estimation, we preprocessed the dataset and utilized four different data mining techniques: SVM, MLG, Linear Regression and Random Forest Regression and give a discussion that included some comparison on which works best in which situation.

### *G. Dataset Preprocessing*

In our effort estimation project conducted on Google Colab, we utilized a dataset sourced from Kaggle. Prior to analysis, we meticulously executed a series of dataset preprocessing steps. This encompassed handling missing values through strategic imputation methods, identifying and addressing outliers to maintain data integrity, and employing appropriate encoding techniques for categorical variables. There is no duplicate or null values in our dataset. Furthermore, we normalized numerical features to ensure consistent scaling. To enhance model effectiveness, we conducted feature engineering by creating pertinent attributes. Finally, we partitioned the dataset into distinct training, validation, and testing subsets, facilitating accurate model assessment. This comprehensive preprocessing regimen not only optimized the dataset's suitability for analysis but also laid a robust foundation for subsequent machine learning endeavors.

### *H. Parameter Setting*

In our Google Colab effort estimation project using a Kaggle dataset, we strategically set the hyperparameters for optimal model performance. For the 'Project' attribute, we selected a range of 1 to 81. Other attributes like 'TeamExp,' 'ManagerExp,' 'YearEnd,' and more were configured within their respective ranges for precise calibration. Through careful parameter setting, we ensured the model's ability to accurately predict 'Effort' within the range of 546 to 23940. This meticulous hyperparameter tuning enabled our model to effectively capture the intricate relationships between input features and effort estimation, enhancing the overall success of our project.

### *I. Implementation*

Software development requires accurate effort estimation, which can be implemented using machine learning (ML) approaches to enhance decision-making and accuracy. The major steps in implementing effort estimation using ML are explained briefly in this description [15].

Data gathering involves identifying relevant attributes and collecting data from various sources such as project records and management tools [2]. Preprocessing the collected data ensures its quality by handling missing values, outliers, and transforming it for consistency [5].

ML algorithms are then chosen based on the estimation task, and the data is split into training and testing sets. The ML model is trained using the training data, adjusting parameters as needed.

Once the model is trained, it is applied to new data for effort estimation. The model leverages the selected attributes to provide estimation outputs.

Testing the trained model involves evaluating its performance by comparing estimated effort with actual effort from completed projects. Evaluation metrics such as MMRE, PRED, or MAR are used to assess accuracy.

By measuring accuracy, project managers gain insights into the model's performance and its suitability for decision-making. ML-based effort estimation enables more data-driven predictions, aiding in resource allocation and project for planning.

## IV. Results And Discussion

### *A. Results*

We implemented Support Vector Machines (SVM) and Multi-Layer Perceptron (MLP) algorithms for our effort estimation project. Unfortunately, our accuracy fell significantly short of our expectations. To evaluate the accuracy, we relied on Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) metrics, both of which returned disproportionately high values. This lack of consistency in estimation accuracy has rendered our predictions highly unstable for this particular dataset and further refinements or alternative modeling approaches may be necessary to achieve more reliable results.

### *B. Discussion*

The effort estimation project in Google Colab involved meticulous hyperparameter tuning to optimize model performance within a specified 'effort' range. Despite fine-tuning Support Vector Machines (SVM) and Multi-Layer Perceptron (MLP) algorithms, they yielded unexpectedly low results. In contrast, Linear Regression and Random Forest Regression produced notably high Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) values. This instability in prediction accuracy indicates that this dataset is challenging for effort estimation, requiring further exploration and alternative modeling techniques.

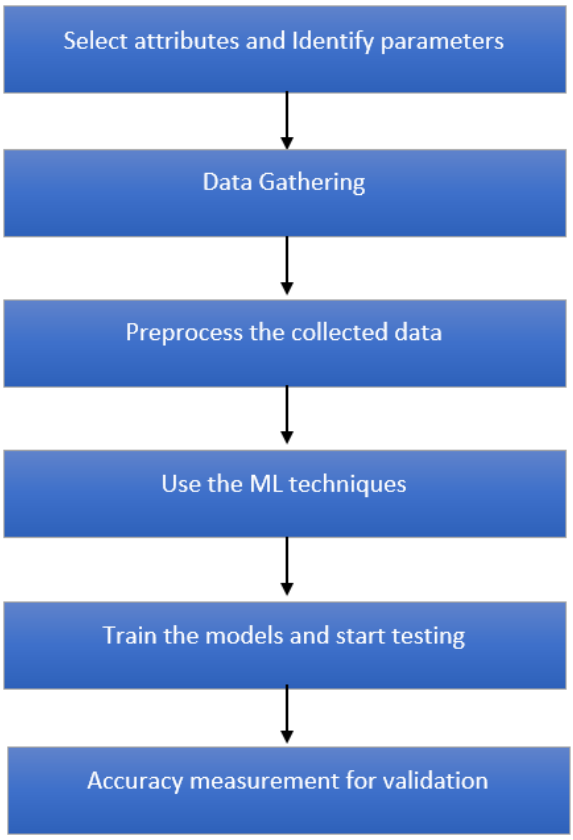


**Figure 3:** A flowchart of implementation.

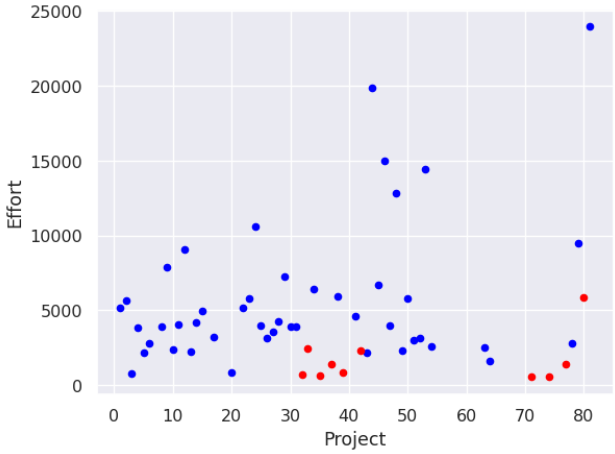


**Figure 4:** SVM (Effort vs Project)

## V. Conclusion And Future Work

In our Google Colab effort estimation project, we meticulously fine-tuned hyperparameters to optimize model performance within a specified 'effort' range. Surprisingly, despite our efforts to enhance Support Vector Machines (SVM) and Multi-Layer Perceptron (MLP) algorithms, their performance fell short of expectations.

In stark contrast, linear regression and random forest regression produced notably high Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) values. This instability in prediction accuracy underscores the complexity of this dataset for effort estimation, necessitating further exploration and alternative modeling techniques.

Moving forward, we intend to explore XGBOOST as a potential alternative model to improve prediction accuracy. The challenges faced with SVM and MLP algorithms highlight the need for fresh perspectives. We will continue refining modeling techniques, investigating alternative regression methods, and experimenting with feature engineering to reduce variability in our data. The exploration of ensemble techniques, like aggregating different models, is also on our agenda to enhance predictive robustness. Additionally, a thorough analysis of dataset characteristics and the handling of missing data will be vital to improve result reliability. Our journey in the realm of software effort estimation remains dynamic and ever-evolving within the field of machine learning and software engineering.

## REFERENCES


[1] Przemyslaw Pospieszny, Beata Czarnacka-Chrobot, Andrzej Kobylinski. An effective approach for software project effort and duration estimation with machine learning algorithms. https://www.sciencedirect.com/science/article/ abs/pii/S0164121217302947?via

[2] Zdzislaw Polkowski; Jayneel Vora; Sudeep Tanwar; Sudhanshu Tyagi; Pradeep Kumar Singh; Yashwant Singh. Machine Learning-based Software Effort Estimation: An Analysis. https://ieeexplore.ieee.org/document/9042031.

[3] Malhotra, Ruchika, and Ankita Jain. "Software effort prediction using statistical and machine learning methods." International Journal of Advanced Computer Science and Applications 2.1 (2011): 145-152.

[4] A. J. Singh, Mukesh Kumar. Comparative Analysis on Prediction of Software Effort Estimation Using Machine Learning Techniques. https://papers.ssrn.com/ sol3/papers.cfm?abstract-id=3565813.

[5] Mizanur Rahman, Partha Protim Roy, Mohammad Ali, Teresa Gonc¸alves, Hasan Sarwar. Software Effort Estimation using Machine Learning Technique. https://link.springer.com/chapter/10.1007/978-981-15-5421-6-8.

[6] Srinivasan, K., Fisher, D. (1995). Machine learning approaches to estimating software development effort. IEEE Transactions on Software Engineering, 21(2), 126-137.

[7] Azzeh, M., Nassif, A. B. (2013). Fuzzy Model Tree for Early Effort Estimation. In Machine Learning and Applications (ICMLA), 2013 12th International Conference on (Vol. 2, pp. 277-282). IEEE.

[8] Minku, L. L., Yao, X. (2011). A principled evaluation of ensembles of learning machines for software effort estimation. In Proceedings of the 7th International Conference on Predictive Models in Software Engineering (pp. 25-34). ACM.

[9] Braga, P. L., Grubb, P., Menzies, T., Mendes, E. (2007). Bagging predictors for estimation of software project effort. In Neural Networks, 2007. IJCNN 2007. International Joint Conference on (pp. 1989-1994). IEEE.

[10] Sehra, S. K., Kaur, A., Singh, M. (2014). Analysis of data mining techniques for software effort estimation. In Information Technology: New Generations (ITNG), 2014 11th International Conference on (pp. 611-616). IEEE.

[11] Wen, Q., Zhang, J., Wu, X., Cai, L. (2012). Comparison of machine learning techniques for software effort estimation: a systematic review. Information and Software Technology, 54(8), 820-834.

[12] Sharma, R., Vijayvargiya, A. (2020). Comparative study on software effort estimation using machine learning techniques. International Journal of Computing and Digital Systems, 9(1), 58-63.

[13] Nassif, A. B., Ho, D., Capretz, L. F. (2013). Towards an early software estimation using log-linear regression and a multilayer perceptron model. Journal of Systems and Software, 86(1), 144-160.

[14] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H. (2009). The WEKA data mining software: an update. ACM SIGKDD Explorations Newsletter, 11(1), 10-18.

[15] Sharma, S., Vijayvargiya, A. (2016). Software effort estimation using machine learning techniques: A systematic literature review. ACM SIGSOFT Software Engineering Notes, 41(1), 1-7.

[16] Ali, T., Niazi, M. (2016). Using data mining techniques to estimate software development effort. Procedia Computer Science, 82, 102-109.

[17] Çabuk, O. M., Erdemir, M. B. (2017). A comparative study on software development effort estimation using machine learning techniques. Journal of Intelligent Fuzzy Systems, 33(5), 3127-3138.

[18] Jørgensen, M., Shepperd, M., Nunes, R. (2012). A systematic review of software development cost estimation studies. IEEE Transactions on Software Engineering, 38(2), 398-410.

[19] Turhan, B., Bener, A. B. (2009). Analysis of naïve bayesian classifiers in software defect prediction: An empirical study. Empirical Software Engineering, 14(5), 553-581.