International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7624
***Article Received**: 01 June 2023 **Revised**: 29 July 2023 **Accepted**: 09 August 2023*

Numerical Simulation and Design of Ensemble Learning Based Improved Software Development Effort Estimation System

Rajani Kumari Gora[1], Ripu Ranjan Sinha[2]
[1]Research Scholar, Computer Science, S. S. Jain Subodh P.G. College
Rajasthan Technical University, Kota
rajanigora@gmail.com
[2]Professor, Computer Science, S. S. Jain Subodh P.G. College
Rajasthan Technical University, Kota
drsinhacs@gmail.com

Abstract— This research paper proposes a novel approach to improving software development effort estimation by integrating ensemble learning algorithms with numerical simulation techniques. The objective of this study is to design an ensemble learning-based software development effort estimation system that leverages the strengths of multiple algorithms to enhance accuracy and reliability. The proposed system combines the power of ensemble learning, which involves aggregating predictions from multiple models, with numerical simulation techniques that enable the modelling and analysis of complex software development processes. A diverse set of software development projects is collected, encompassing various domains, sizes, and complexities. Ensemble learning algorithms such as Random Forest, Gradient Boosting, Bagging, and AdaBoost are selected for their ability to capture different aspects of the data and produce robust predictions. The proposed system architecture is presented, illustrating the flow of data and components. A model training and evaluation pipeline is developed, enabling the integration of ensemble learning and numerical simulation modules. The system combines the predictions generated by the ensemble models with the simulation results to produce more accurate and reliable effort estimates. The experimental setup involves a comprehensive evaluation of the proposed system. A real-world dataset comprising historical project data is utilized, and various performance metrics, including Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), are employed to assess the effectiveness of the system. The results and analysis demonstrate that the ensemble learning-based effort estimation system outperforms traditional techniques, showcasing its potential to enhance project planning and resource allocation.

Keywords- Software Development, Effort Estimation, Stacking, MAE, RMSE, R2 Score.

I. INTRODUCTION

Software Effort Estimation (SEE) is one of the key tasks in Software Project Management (SPM). It is the process of determining the total amount of effort required to develop a software product before initiation. The main advantage of effort estimation is that it helps plan and predict the way to implement software projects. However, it has been considered one of the most difficult tasks as it plays a vital role in determining the success of the software project. Moreover, it is essential for being competitive in the market. Possibly, accurate prediction helps the Project Managers (PMs) utilise the available resources more efficiently and complete the project within the scheduled deadline without exceeding the budget. However, overestimating leads to the failure of the software project by underutilizing valuable resources and losing bidding. Similarly, underestimating causes tighter schedule deadlines, more defects, and a loss of software quality. So, it is important to predict the software effort more accurately and precisely. However, it has been found that over 63 percent of the project budgets are higher than the initial estimates. Hence, Software Effort estimation remains one of the toughest tasks in the software industry.

Many methods have been developed and used in estimating the total effort. Generally, effort is commonly estimated using a technique known as, man-power loading. According to this technique, the effort is expressed as a function of number of engineering ways and management approaches required to develop a software product. At the initial stage of effort estimation, it is important to consider the project feasibility as the data's collected during the initial period of a software life-cycle is imprecise and unsure. Although accurate prediction is critical, it is extremely difficult to determine in the initial stage. Typically, effort estimation is done using the four activities shown in Figure 1.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7624
***Article Received**: 01 June 2023 **Revised**: 29 July 2023 **Accepted**: 09 August 2023*

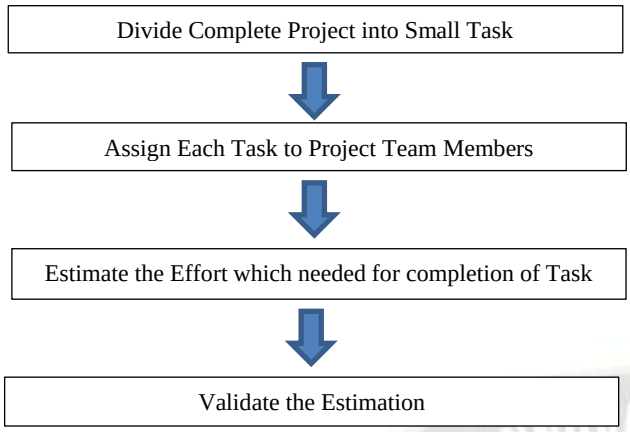

Figure 1. Estimation approaches

As shown in Figure 1, the project managers divide the whole project into smaller modules which is further divided into sub-modules during the requirement analysis of the project. Then, they form project teams satisfying all criteria for the project with experienced members. Each module of the project is assigned to specific team, and they compute the effort needed to complete the module. Based on the effort required for a module, the overall budget is estimated.

This study explores a variety of estimating methods, including Machine Learning, algorithmic models, Function Point, Top-Down, Bottom-Up, and Expert estimation. In the modern world, neural networks, COCOMO, and planning poker are frequently used.

Researchers have looked into the application of machine learning techniques in effort estimation to address these limitations. In order to anticipate the amount of work required, machine learning algorithms can analyse historical project data, spot patterns, and gain expertise from previous projects. Recent years have seen an increase in interest in ensemble learning, a method that integrates numerous models to create predictions. By combining the advantages of various models and minimising the weaknesses of individual algorithms, ensemble learning has proven to increase accuracy and robustness.

In addition to machine learning, numerical simulation techniques have shown promise in modeling and analyzing complex systems. Simulation allows for the representation of real-world processes, considering uncertainties and variations that occur during project execution. By simulating the software development process, it becomes possible to generate insights into the potential outcomes, identify bottlenecks, and evaluate different scenarios.

This research paper proposes a novel approach to improve software development effort estimation by integrating ensemble learning algorithms with numerical simulation techniques. The objective is to design an ensemble learning-based effort estimation system that leverages the power of multiple algorithms and incorporates the dynamic nature of software development processes.

The primary aim of this research is to enhance the accuracy and reliability of effort estimation, ultimately leading to better project planning and resource allocation. By combining ensemble learning algorithms with numerical simulation, the proposed system seeks to address the limitations of traditional estimation techniques and provide more robust predictions.

To achieve this objective, a comprehensive methodology will be developed. This methodology will involve collecting and preprocessing a diverse set of software development project data, encompassing various domains, sizes, and complexities. Ensemble learning algorithms such as Random Forest, Gradient Boosting, Bagging, and AdaBoost will be employed to capture different aspects of the data and produce accurate predictions.

Furthermore, the integration of numerical simulation techniques will enable the modeling and analysis of the dynamic aspects of the software development process. Monte Carlo Simulation and Discrete Event Simulation methods will be utilized to consider uncertainties and variations that occur during project execution.

The experimental setup will involve evaluating the proposed system using a real-world dataset comprising historical project data. Various performance metrics, including Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), and R^2will be employed to assess the effectiveness of the system in comparison to traditional estimation techniques. The results and analysis will demonstrate the system's ability to improve effort estimation accuracy and reliability.

In conclusion, this research aims to address the challenges of software development effort estimation by proposing a novel approach that combines ensemble learning algorithms with numerical simulation techniques. The integration of these techniques seeks to enhance accuracy, robustness, and the ability to capture uncertainties in the estimation process. The outcomes of this research will contribute to better project planning and resource allocation in software development projects, ultimately improving the success rates and outcomes of software development endeavors.

II. RELATED WORKS

Software development effort estimation (SDEE) is the process of estimating amount of work necessary to construct a software system. SDEE is a subfield of software effort estimation which takes into account estimates for both software development and software maintenance. Researchers sometimes use the words "software cost estimation" and "software effort estimation" interchangeably, despite the fact that effort contributes only a little amount to total software cost. Predicting development

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7624
***Article Received**: 01 June 2023 **Revised**: 29 July 2023 **Accepted**: 09 August 2023*

work properly in early stages of software life cycle is essential for good project management. In SDEE sector, various estimating approaches have been suggested since 1980s. In their study, Jrgensen and Shepperd found that there are as many as 11 different methods for measuring SDEE. It is observed that regression-based methods predominate but utilize of expert opinion and analogy methods is on the rise. Interest in SDEE studies has increased in recent years, particularly in those that employ ML (machine learning) based methodologies. In fact, ML-based methods are regarded by several experts as one of the top 3 estimating methods (other two are expert judgement and algorithmic model).

In field of software development, estimating software projects is the most difficult undertaking. There will be no appropriate planning and management of software development projects if accurate and dependable estimates are not supplied. Even when all relevant elements are considered during software development process, project estimates are not always correct. It does not leverage estimations to improve software development. When a project is underestimated, consequences, including inadequate quality assurance efforts, insufficient manpower, and missed deadlines, result in a loss of confidence [1]. The estimation of a software project is required in order to account for underestimations and overestimations in terms of cost, effort, etc. More resources than a project needs will be employed, driving up product pricing; hence accurate estimation is essential. In spite of vast amount of empirical studies on ML models, contradictory result has been reported on estimation accuracy of ML models, comparisons among non-ML models or ML models, and differences across ML models. For example, it has been pointed out that the accuracy of estimates changes when same ML model is built using many different historical project data sets or experimental designs. Research published claims that the ML model is more effective than regression model, whereas research published concludes that regression approach is more successful. Comparing various ML models (such as ANN and case-based reasoning), research indicated which former outperformed latter, whilst a study found opposite conclusions [2]. The time and energy spent by developers is the most valuable asset for most software projects, and how that asset is divided may have a considerable impact on the project's timeframe and budget. In order to successfully complete a single instance of a project, it is crucial to be able to estimate necessary developer effort in advance and alter these estimates during project's life cycle. Inaccurate (often overoptimistic) work estimates are a common reason for projects to fail [3] have clear ramifications for the organization's overall performance. As a consequence, effort estimating has been subjected to many studies over past few decades, with the aim of evaluating and improving existing estimation methods, as well as developing wholly new ones [4]. Historically, most prevalent estimating technique has been some variation of expert estimation, i.e., relying on human subject matter experts to examine pertinent data and provide an estimate. Moreover, data-driven solutions have been created, although their implementation in practice is uncommon for a variety of reasons. Among these data-driven strategies are several ML algorithms [5]. The application of ML techniques to this issue domain is not a new phenomenon; research in this direction has been undertaken rather continuously over past few decades, far before present AI- related research boom.

In context of software development, estimating is understood to mean making educated guesses about unknown quantities such as money and time spent. Estimate is a crucial aspect of software project management since it impacts both client and developer sides. If the estimate is accurate, development can be planned, progress can be tracked, and customer can negotiate price and completion date. However, as the leading cause of software failure is erroneous estimation of vital characteristics, estimating becomes a crucial and essential duty in predicting the dependability of software [6].

An enhanced method for Scrum-based Agile projects using 36 success indicators, resulting in more cost-effective and efficient outcomes [7]. A utilized pre-trained embedding models to improve textual requirements gathering for effort estimation, achieving reliable and efficient results [8]. Applied machine learning and deep learning techniques to predict hardware development tasks' duration, demonstrating the applicability of software effort estimation in the hardware sector [9]. A TLBO-guided TABE system for more accurate work estimates, achieving reduced mean absolute residue (MAR) and mean magnitude of relative error (MMRE) values [10]. Developed models for agile project time and effort estimation based on initial software requirements, showing improved accuracy when integrating application domain groups and peak employees [11]. Identified causal factors affecting effort estimation and proposed an Extreme Learning Machine (ELM) model with superior performance in software design effort estimation [12]. Artificial Neural Networks (ANN) estimation approaches outperformed conventional methods, highlighting the importance of comparing and contrasting different estimation methodologies [13]. Conducted empirical interviews to investigate estimation methods used by financially sound firms, emphasizing the need for suitable estimation methods and control [14]. Compared various ANN models and found that they provided more accurate estimates compared to traditional approaches such as Use-case methods, Function point, and COCOMO [15]. A continuous simulation to enhance project work estimation, enabling better understanding of effort ranges and risk management decisions [16]. Developed a non-linear approach using Multi-Layer Perceptron’s for effort

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7624
***Article Received**: 01 June 2023 **Revised**: 29 July 2023 **Accepted**: 09 August 2023*

estimation, showing improved performance compared to linear regression [17]. Conducted a literature review on ML-based software effort estimation, highlighting the trends and methodologies employed in the field [18]. To design mobile applications based on accurate effort estimations, focusing on early estimation techniques and efficient resource allocation [19]. A fuzzy approach to estimate industrial effort when conventional data analysis methods were not feasible [20]. In summary, these studies have contributed to the field of software development effort estimation by proposing enhanced methods, leveraging machine learning and deep learning techniques, exploring estimation in different domains, and emphasizing the importance of accurate and efficient estimation approaches.

III. METHODOLOGY

The experiment's goal is to combine the several independent machine learning techniques to create an ensemble model using the stacking generalisation ensemble method. The necessary libraries and functions were imported. After that, preprocessing was done on the datasets. The ensemble model is constructed, and all of the approaches are set up. The model is then trained and tested after that. Finally, the test findings are assessed.

The below figure 2 shows steps involved in preprocessing, training and testing stages of the experiment.

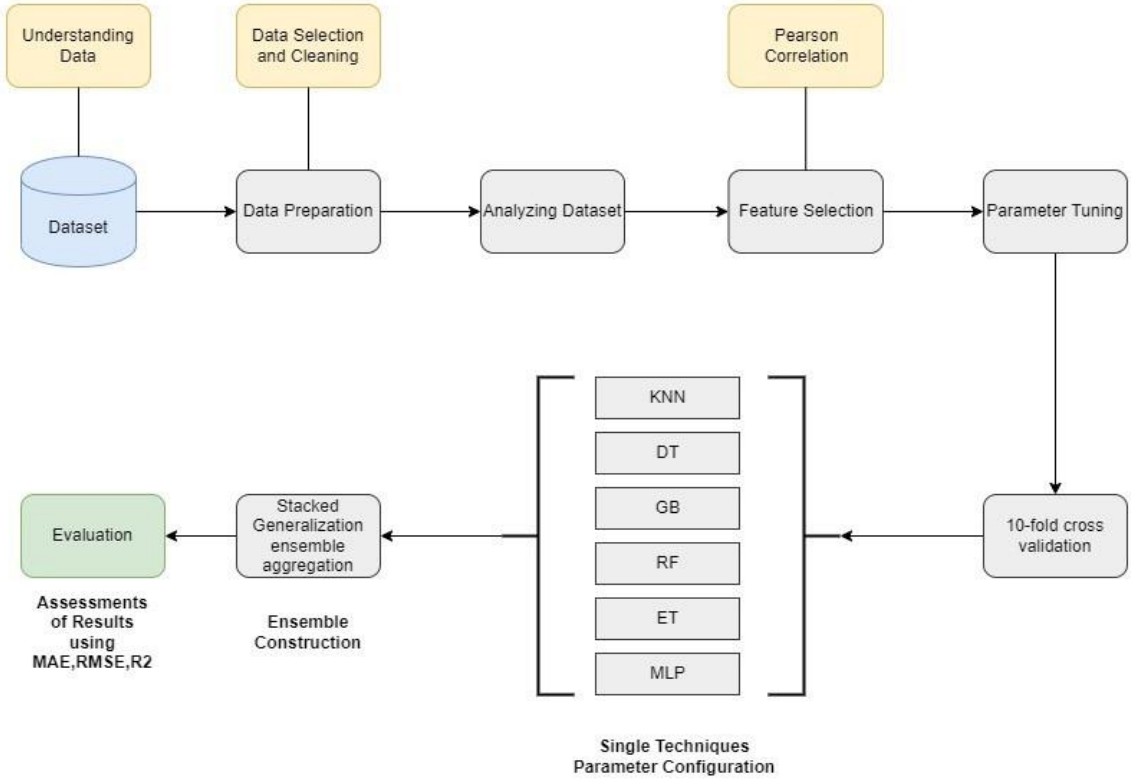

Figure 2. Description of methodology

3.1 Machine Learning Methods

The following machine learning methods are used to a common data set to measure how accurately defects are predicted and how much effort would be required to develop high-quality software. The decision was made to employ any machine-learning approach in the proposed research based on the literature review. Several researchers have previously employed some of the following machine learning approaches for their research. Unfortunately, none of these methods have previously been utilized to predict faults and estimate work using sophisticated machine learning techniques. Each proposed contribution gives an in-depth analysis of the results obtained through the application of these methods to the provided dataset.

3.1.1 Linear Regression (LR)

LR Linear regression (LR) is a type of predictive analysis used to discover the association between two continuous variables. It involves a statistical relationship rather than a deterministic one. The first type of regression analysis used in practical applications focuses on the probability distribution of the

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7624
***Article Received**: 01 June 2023 **Revised**: 29 July 2023 **Accepted**: 09 August 2023*

response variable. LR is represented by a simple equation with one dependent variable [9], as shown in eq.

$$\rho = \beta_0 + \beta_1 + \gamma \quad (1)$$

Here, ρ represents the predicted response variable, and γ is the independent variable score. LR analysis is utilized in various research domains, such as trend forecasting, determining the strength of predictors, and predicting the effects.

3.1.2 Multiple Linear Regressions

Multiple linear regression is a statistical technique used to predict the value of a response variable by establishing a linear relationship between the dependent (response) and independent (predictor) variables. This regression model involves more than one regression variable, and the independent variables are not necessarily positively correlated. The equation for multiple linear regression is:

$$\gamma = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \beta_3 * x_3 + \cdots + \beta_n * x_n \quad (2)$$

Here, x represents the independent variables, β_0 is the intercept, and γ is the dependent variable.

3.1.3 Multi-Layer Perceptron (MLP)

A multi-layer perceptron is a preceptor made up of several layers, including an input layer, a yield layer, and a veiled layer, which contain the information sources and loads needed to start any hub. A series of highlights are performed by the veiled layer. Any problem can be solved with two buried layers, while highlights imply that more levels could be preferable. Through the yield layer, MLP can be configured to determine how to transform input data into a supported response. To lessen the assessment of the error job, the count modifies the heaps of each affiliation. Following a sufficient number of planning cycles in which this cycle is repeated, the association will typically join a state where the error of the assessment is minimal. The botched work related to the association loads is corrected, and the heaps are then modified to lessen the mistake. The confusion is represented by the squared Euclidean detachment between the actual yield and the required yield indicated in Eq.

$$\triangle\omega = y * d * x \quad (3)$$

d - Predicted output

y - Learning rate, usually less than 1

x -Input data

3.1.4 Random Forest (RF)

Random Forest (RF) is a machine learning method used for regression and classification tasks. During training, an RF builds a collection of decision trees and combines their outputs to provide a class or regression value. RF is an ensemble of tree predictors, where each tree predicts the target variable based on a random vector sampled independently. The principle behind RF is that a group of weak learners can combine to form a strong learner. It helps address overfitting issues and produces accurate classifiers and regressors [10].

3.2 Need for Intelligent ML Methods over Classical Methods

Based on the "no free lunch theorem" (NFL), a single machine learning classification algorithm is not the ideal solution for Software Product Development (SPD). So, it is necessary to apply intelligent techniques for SPD in this manner. In the last few decades, several traditional software development approaches and measurement techniques have been suggested for effort estimate. These models may provide useful growth forecasts for reliable programming and its development. All existing prediction and effort models are regarded as outstanding. However, the solution is still in its infancy; standard effort models, and effort estimation are the appropriate methods for determining the quality of a product. The estimation of software efforts are essential for avoiding surprising results and identifying development requirements. If developers are able to predict these components accurately, they will use new intelligent ML methods for delivering reliable and high- quality software products in favor of statistically-based approaches. Using ensemble learning and intelligent boosting methods to estimate effort in this scenario.

3.2.1 Ensemble Learning

Ensemble learning is an emerging field that combines various aspects of developmental math, machine learning, and probabilistic thinking. It employs multiple models to enhance the stability and predictive capabilities of a single model. By leveraging the strengths of individual models, ensemble learning techniques outperform standalone models in making accurate predictions.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7624
Article Received: 01 June 2023 Revised: 29 July 2023 Accepted: 09 August 2023

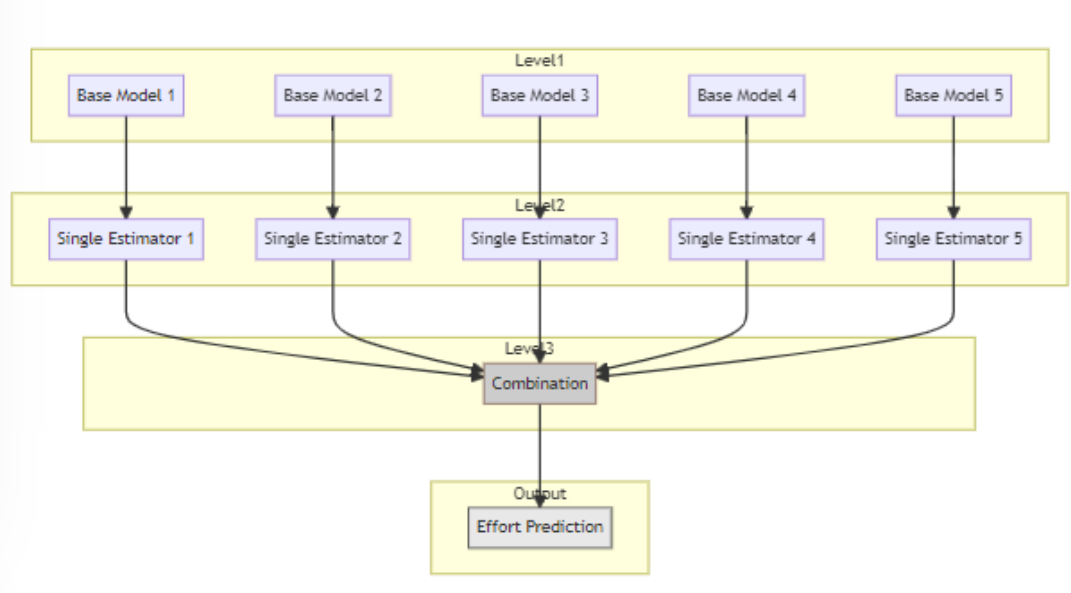

Figure 3. Ensemble learning model

Figure 3 visually depicts training data from the original dataset, a base model (decision tree) used to predict residuals, and the amalgamation of multiple machine learning base classifiers to create an ensemble learner that surpasses the performance of each individual classifier. It is widely acknowledged that the base classifiers should be both diverse and accurate. Having substantial and uncorrelated differences in errors among the base classifiers allows for error compensation between them.

3.2.2 Bagging, Boosting and Stacking

Bagging (Bootstrap Aggregating) and Gradient Boosting are two distinct ensemble learning techniques used in machine learning. They can be used separately or in combination, but they have different principles and objectives.

1. Bagging: Bagging is an ensemble learning technique that aims to reduce the variance of a predictive model. It involves creating multiple subsets of the original training dataset through a process called bootstrap sampling. Bootstrap sampling involves randomly sampling the original dataset with replacement, which means that each subset can contain duplicate instances and some instances may be excluded. These subsets are then used to train multiple base models, such as decision trees, independently and in parallel. Once the base models are trained, predictions are made by aggregating the predictions of each model. For regression problems, the predictions are typically averaged, while for classification problems, majority voting is often used. By combining the predictions of multiple models, bagging can reduce the impact of outliers and noise in the data, leading to improved overall performance and generalization.
2. Gradient Boosting: Gradient Boosting is an ensemble learning technique that focuses on building a strong predictive model by iteratively combining weak models. Unlike bagging, gradient boosting is a sequential process that builds models in a stage-wise manner. It aims to minimize a loss function by adding weak models to the ensemble, with each model correcting the mistakes made by the previous models. The process begins by training an initial weak model, usually a decision tree. The errors or residuals from this model are then calculated, representing the difference between the predicted and actual values. The subsequent weak models are trained to predict these residuals. In each iteration, the model is fit to the negative gradient of the loss function, hence the name "gradient boosting." The predictions from all the weak models are then combined to obtain the final prediction. To prevent over fitting, regularization techniques, such as shrinkage/learning rate and subsampling, are often used.
3. Stacking: Stacking is a Heterogeneous Ensemble learning technique. It combines predictions from multiple machine learning algorithms and uses these predictions as inputs to second-level learning models (Meta model) to provide final prediction. Stacking, also known as stacked generalization, It leverages the principle of model aggregation, where the predictions from diverse models are blended together to exploit their complementary strengths and mitigate individual model weaknesses.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7624
***Article Received**: 01 June 2023 **Revised**: 29 July 2023 **Accepted**: 09 August 2023*

3.2.3 Combining Bagging and Gradient Boosting

It is possible to combine bagging and gradient boosting to create a powerful ensemble model. This technique, known as bagging with gradient boosting, involves applying bagging to the individual stages of the gradient boosting process. Instead of training a single weak model at each stage, multiple weak models are trained using bootstrap samples of the data. The predictions of these models are then combined to form the ensemble prediction.

Bagging with gradient boosting can provide additional benefits by reducing over fitting and improving the stability of the gradient boosting process. It can also help in capturing different sources of variability in the data and enhancing the model's ability to generalize well.

In summary, bagging and gradient boosting are both ensemble learning techniques, but they have different objectives and operate in distinct ways. Bagging reduces variance by training multiple models independently and combining their predictions, while gradient boosting builds a strong model by iteratively correcting errors using weak models. Bagging with gradient boosting combines these two techniques to create a more robust and accurate ensemble model.

The current study focuses on techniques and methods for assessing out how much work needs to be done, which helps figure out how well a software product works [11]. With that in mind, this thesis shows new ways to predict effort, use the ensemble machine learning model, and improve the presentation and importance of the measure by similarity strategy.

3.2.4 Evaluation Criteria

The exhibition evaluation acquired utilizing different ensemble methods has been completed using various rules. To find which model/method is better, the accuracy of the models has be calculated. The estimation models is evaluated using Mean Absolute Error (MAE), Root mean square error (RMSE) and R squared score (R2 Score) metrics.

3.2.4.1 Mean Absolute Error (MAE)

MAE, or Mean Absolute Error, is a commonly used metric to evaluate the accuracy of software development effort estimation models. It measures the average magnitude of the errors between the predicted and actual effort values. The lower the MAE, the more accurate the estimation model.

Mathematically, MAE is defined as:

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n} \quad (4)$$

Where:

- MAE is the Mean Absolute Error.
- n is the total number of estimation instances.
- Yi represents the predicted effort values.
- Xi represents the actual effort values.

MAE calculates the absolute difference between the predicted and actual effort for each estimation instance and then takes the average of these differences.

By using MAE and statistical significance testing, we can evaluate the accuracy of estimation models and compare them to make informed decisions in software development effort estimation.

3.2.4.2 Root Mean Square Error (RMSE)

RMSE, or Root Mean Square Error, is another commonly used metric to evaluate the accuracy of software development effort estimation models. It measures the average magnitude of the squared errors between the predicted and actual effort values. Similar to MAE, a lower RMSE indicates a more accurate estimation model.

Mathematically, RMSE is defined as:

RMSE = √((1/n) * Σ(Y_pred - Y_actual)^2) (5)

Where:

- RMSE is the Root Mean Square Error.
- n is the total number of estimation instances.
- Y_pred represents the predicted effort values.
- Y_actual represents the actual effort values.

RMSE calculates the squared difference between the predicted and actual effort for each estimation instance, takes the average of these squared differences, and then takes the square root of the result.

We would collect the RMSE values for each estimation model on the same set of estimation instances and then perform the paired t-test. The resulting p-value will indicate whether the observed difference in RMSE values is statistically significant.

RMSE, along with statistical significance testing, helps in evaluating the accuracy of estimation models and comparing them to make informed decisions in software development effort estimation.

3.2.4.3 R^2 Score

R^2 Score, also known as the coefficient of determination, is a metric used to assess the goodness of fit of a regression model in software development effort estimation. It indicates the proportion of the variance in the dependent variable (effort values) that can be explained by the independent variables (predictors) in the model. R2 Score ranges from 0 to 1, where 1 represents a perfect fit.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7624
***Article Received**: 01 June 2023 **Revised**: 29 July 2023 **Accepted**: 09 August 2023*

Mathematically, R^2 Score is defined as:

$$R2 = 1 - (SSres / SStot) \quad (6)$$

Where:

- R^2 is the coefficient of determination (R2 Score).
- SSres is the sum of squared residuals or errors.
- SStot is the total sum of squares.

SSres measures the discrepancy between the predicted effort values and the actual effort values. It is calculated by summing the squared differences between each predicted effort value and its corresponding actual effort value. SStot represents the total variation in the actual effort values. It is calculated by summing the squared differences between each actual effort value and the mean of all actual effort values.

IV. RESULTS AND DISCUSSION

The goal of the project is to develop an effective effort estimation model on China Dataset with the different machine learning methods for achieving best possible accuracy level, optimizing software projects by estimating efforts for the same using machine learning techniques. The step-by-step process is as follows-

This data collection that is the newest dataset in the area of effort estimation contains information on 499 software projects with 18 features belonging to various software companies and firms. The index description of China dataset are as follows-

- ID
- AFP
- Input
- Output
- Enquiry
- File
- Interface
- Added
- Changed
- Deleted
- PDR_AFP
- PDR_UFP
- NPDR_AFP
- NPDU_UFP
- Resource
- Dev.Type
- Duration
- N_effort
- Effort

Different Models has been implemented on China Dataset which are enlisted as follows

- K-Neighbors
- Decision Tree
- Gradient Boosting
- Random Forest
- Extra Trees
- MLP

Effort estimation in software development projects is a crucial task that influences project planning, resource allocation, and project management. This paper focuses on exploring various machine learning algorithms for software development effort estimation, specifically k-neighbors, decision tree, gradient boosting, random forest, extra trees and multilayer perceptron (MLP). The paper provides a comprehensive analysis of these methods using equations, tables, and a comparative assessment to evaluate their performance in ensemble-based effort estimation.

In the tuned models following methodologies have been applied to improve the accuracy and performance of implemented models. The improved machine learning methodologies has been applied and implemented using following steps-

- Bagging
- Adaboost
- Stacking

Ensemble methods are powerful techniques in machine learning that combine multiple individual models to improve predictive performance. Three commonly used ensemble methods are bagging and AdaBoost, and Stacking.

In summary, bagging, AdaBoost, and stacking are all ensemble methods that combine multiple models to improve predictive performance. Bagging focuses on reducing variance and improving stability, AdaBoost emphasizes weak classifiers' correct predictions by adjusting sample weights, and stacking combines diverse models through a meta-model to leverage their collective strengths. These methods have been widely applied in various machine learning tasks, demonstrating their effectiveness in boosting performance and handling complex data patterns.

Training and testing data with K-Fold Cross Validation

To create model 80% training data set and 20% testing data sets. We split the dataset using K-fold cross validation. In this approach, the data set is divided into k folds, or smaller subsets, and then trained on all but one (k-1) of them before the trained model is evaluated. In this approach, we iterate k times, each time reserving a new subset for testing.

Table 1. Analysis of performance parameters with bagging ensemble machine learning methodologies.

Methodology	MAE	RMSE	R2_Score
K- Neighborhood	0.7046	0.2417	0.8051
Decision Tree	0.5591	0.1878	0.8823
Gradient Boosting	0.5828	0.1925	0.8709
Random Forest	0.6490	0.2089	0.8508
Extra Trees	0.5152	0.2017	0.8646
MLP Tuning	0.4003	0.1388	0.9268
SVR	0.7141	0.3442	0.61
Linear Regression	0.4453	0.1295	0.94

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7624
***Article Received**: 01 June 2023 **Revised**: 29 July 2023 **Accepted**: 09 August 2023*

Table 2. Analysis of performance parameters with adaboost machine learning methodologies.

Methodology	MAE	RMSE	R2_Score
K- Neighborhood	0.8434	0.1813	0.8906
Decision Tree	0.801170000	0.2378	0.8117
Gradient Boosting	0.740237124	0.1870	0.8576
Random Forest	0.611473143	0.1993	0.8678
Extra Trees	0.7957723	0.2068	0.8567
MLP	0.836971437	0.1813	0.8906
SVR	0.71412	0.2432	0.79
Linear Regression	0.44543	0.1292	0.94

Table 3. Analysis of performance parameters with stacking machine learning methodology.

Methodology	MAE	RMSE	R2_Score
Stacking	0.5163	0.1586	0.9163

Table 4. Comparative analysis of ensemble methods on dataset.

Ensemble Method	Bagging			Adaboost			Stacking		
Methodology	**MAE**	**RMSE**	**R2_Score**	**MAE**	**RMSE**	**R2_Score**	**MAE**	**RMSE**	**R2_Score**
K-Neighborhood	0.70463	0.241787	0.8051	0.843409	0.1813536	0.8906	0.51627	0.158620	0.9163
Decision Tree	0.55992	0.187894	0.8823	0.801170	0.2378800750	0.8117			
Gradient Boosting	0.58282	0.192577	0.8709	0.740237	0.1870145038	0.8576			
Random Forest	0.64903	0.208941	0.8508	0.611473	0.1993631795	0.8678			
Extra Trees	0.51524	0.201757	0.8646	0.795772	0.2068641	0.8567			
MLP	0.40037	0.138812	0.9268	0.836971	0.1813673	0.8906			
SVR	0.7141	0.3442	0.61	0.71412	0.2432	0.79			
Linear Regression	0.4453	0.1295	0.94	0.44543	0.1292	0.94			

The table provided summarizes the performance of different ensemble methods, namely Bagging and AdaBoost, and Stacking, on various regression tasks. The table includes evaluation metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared Score (R2_Score) for each ensemble method and specific models used within those methods.

V. CONCLUSIONS

Analysing the values in the table, we can observe the performance of different models within each ensemble method. For example, within the Bagging ensemble method, the K-Neighborhood model achieves an MAE of 0.70463, an RMSE of 0.241787, and an R2_Score of 0.8051. Similarly, other models such as Decision Tree, Gradient Boosting, Random Forest, Extra Trees, and MLP Tuning are evaluated within their respective ensemble methods. Comparing the metrics across ensemble methods, we can assess their relative performance. For instance, we can compare the MAE, RMSE, and R2_Score values of the K-Neighborhood model in Bagging and AdaBoost, Stacking. Additionally, comparing the performance of different models within each ensemble method allows us to understand which models are more effective for the given regression tasks. Overall, the table provides a comprehensive overview of the performance of different ensemble methods and models in terms of MAE, RMSE, and R2_Score. It enables us to compare and assess the effectiveness of the ensemble methods and individual models in making accurate predictions for the given regression tasks. Such analysis can guide the selection of the most suitable ensemble method or model for future regression problems and inform decision-making in real-world applications. The comparative analysis suggests that ensemble methods improve the performance of machine learning methods with respect to base models. In this research different approaches have been presented to improve the performance of machine learning methodologies for SDEE. These methodologies include hyper parameter tuning, ensemble methods such as stacking, adaboost and bagging. The proposed methodologies have been applied on regressor methods and tested on datasets. The improved performance parameter

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10s
DOI: https://doi.org/10.17762/ijritcc.v11i10s.7624
***Article Received**: 01 June 2023 **Revised**: 29 July 2023 **Accepted**: 09 August 2023*

indicates and validates the effectiveness of ensemble methods. The obtained R-2 score was over 0.9163 for china dataset which indicates the improved performance of the proposed methodologies. The overall analysis and comparative assessment of the machine learning algorithms, hyperparameter tuning, and ensemble methods demonstrated the effectiveness of ensemble-based effort estimation for software development projects. By leveraging the strengths of different algorithms and combining their predictions, we achieved higher accuracy and improved project planning and management. In conclusion, this project contributed to the field of software development effort estimation by exploring and comparing various machine learning algorithms, optimizing their performance through hyperparameter tuning, and leveraging ensemble methods for improved accuracy. The findings can assist practitioners in choosing the most suitable approach based on their project requirements and data characteristics. Further research can focus on hybrid approaches, domain-specific features, and emerging techniques to enhance effort estimation in software development.

REFERENCES

[1] S. Dragicevic, S. Celar, And M. Turic,-Bayesian Network Model For Task Effort Estimation In Agile Software Development, J. Syst. Softw., (2017), Doi: 10.1016/J.Jss.2017.01.027.

[2] J. Wen, S. Li, Z. Lin, Y. Hu, And C. Huang, -Systematic Literature Review Of Machine Learning Based Software Development Effort Estimation Models, Information And Software Technology. (2012). Doi: 10.1016/J.Infsof.2011.09.002.

[3] P. Pospieszny, B. Czarnacka-Chrobot, And A. Kobylinski, -An Effective Approach For Software Project Effort And Duration Estimation With Machine Learning Algorithms, J. Syst. Softw., (2018), Doi: 10.1016/J.Jss.2017.11.066.

[4] A. Trendowicz, J. Münch, And R. Jeffery, "State Of The Practice In Software Effort Estimation: A Survey And Literature Review," (2011). Doi: 10.1007/978-3-642-22386-0_18.

[5] M. Vyas And N. Hemrajani, -Predicting Effort Of Agile Software Projects Using Linear Regression, Ridge Regression And Logistic Regression, Int. J. Tech. Phys. Probl. Eng., (2021).

[6] P. L. Braga, A. L. I. Oliveira, And S. R. L. Meira, -Software Effort Estimation Using Machine Learning Techniques With Robust Confidence Intervals, Pp. 352– 357, (2008), Doi: 10.1109/His.2007.56.

[7] N. Govil And A. Sharma, -Estimation Of Cost And Development Effort In Scrum-Based Software Projects Considering Dimensional Success Factors, Adv. Eng. Softw., Vol. 172, P. 103209, (2022), Doi: Https://Doi.Org/10.1016/J.Advengsoft.2022.103209.

[8] E. M. De Bortoli Fávero, D. Casanova, And A. R. Pimentel, -Se3m: A Model For Software Effort Estimation Using Pre-Trained Embedding Models, Inf. Softw. Technol., Vol. 147, P. 106886, (2022), Doi: Https://Doi.Org/10.1016/J.Infsof.2022.106886.

[9] H.-C. Jang And S.-C. Wu, -Tracking Of Hardware Development Schedule Based On Software Effort Estimation, In 2022 Ieee 13th Annual Information Technology, Electronics And Mobile Communication Conference (Iemcon), Pp. 305–310 (2022). Doi: 10.1109/Iemcon56893.2022.9946524.

[10] P. Manchala And M. Bisi, -Ensembling Teaching-Learning-Based Optimization Algorithmwith Analogy-Based Estimation To Predict Software Development Effort, In 2022 13th International Conference On Computing Communication And Networking Technologies (Icccnt), Pp. 1–7. (2022) Doi: 10.1109/Icccnt54827.2022.9984558.

[11] W. Rosa, B. K. Clark, R. Madachy, And B. W. Boehm, -Empirical Effort And Schedule Estimation Models For Agile Processes In The Us Dod, Ieee Trans. Softw. Eng., (2022), Doi: 10.1109/Tse.2021.3080666.

[12] H. D. P. De Carvalho, R. Fagundes, And W. Santos, -Extreme Learning Machine Applied To Software Development Effort Estimation, Ieee Access, (2021), Doi: 10.1109/Access.2021.3091313.

[13] L. Lazic, -Artificial Neural Network Architectures And Orthogonal Arrays In Estimation Of Software Projects Efforts Estimation : Plenary Talk, In 2021 Ieee 19th International Symposium On Intelligent Systems And Informatics (Sisy), Pp. 13–14. (2021) Doi: 10.1109/Sisy52375.2021.9582466.

[14] T. Vera, S. F. Ochoa, And D. Perovich, Development Effort Estimation Practices In Small Software Companies: An Exploratory Study, (2020). Doi: 10.1109/Sccc51225.2020.9281161.

[15] P. S. Kumar, H. S. Behera, K. Anisha Kumari, J. Nayak, And B. Naik, -Advancement From Neural Networks To Deep Learning In Software Effort Estimation: Perspective Of Two Decades, Computer Science Review. (2020). Doi: 10.1016/J.Cosrev.2020.100288.

[16] J. Z. Gomes, J. L. Montenegro, J. V. Canto Dos Santos, J. L. V. Barbosa, And C. A. Costa, -A Strategy Using Continuous Simulation To Mitigate Effort Estimation Risks In Software Projects, Ieee Lat. Am. Trans., (2019), Doi: 10.1109/Tla.2019.8932373.

[17] S. Goyal And P. K. Bhatia, -A Non-Linear Technique For Effective Software Effort Estimation Using Multi-Layer Perceptrons, In 2019 International Conference On Machine Learning, Big Data, Cloud And Parallel Computing (Comitcon), Pp. 1–4. (2019) Doi: 10.1109/Comitcon.2019.8862256.

[18] P. Sharma And J. Singh, -Systematic Literature Review On Software Effort Estimation Using Machine Learning Approaches, (2018). Doi: 10.1109/Icngcis.2017.33.

[19] G. Catolino, -Effort-Oriented Methods And Tools For Software Development And Maintenance For Mobile Apps, (2018). Doi: 10.1145/3183440.3183457.

[20] A. Saini, L. Ahuja, And S. K. Khatri, -Effort Estimation Of Agile Development Using Fuzzy Logic, (2018). Doi: 10.1109/Icrito.2018.8748381.

[21] Bhawana Verma, S. K.A. (2019). Design & Analysis of Cost Estimation for New Mobile-COCOMO Tool for Mobile

Application. International Journal on Recent and Innovation Trends in Computing and Communication, 7(1), 27–34. https://doi.org/10.17762/ijritcc.v7i1.5222

[22] S. K.A., Raj, A. ., Sharma, V., & Kumar, V. (2022). Simulation and Analysis of Hand Gesture Recognition for Indian Sign Language using CNN. International Journal on Recent and Innovation Trends in Computing and Communication, 10(4), 10–14. https://doi.org/10.17762/ijritcc.v10i4.5556.

[23] Najneen Qureshi, Manish Kumar Mukhija and Satish Kumar, "RAFI: Parallel Dynamic Test-suite Reduction for Software", New Frontiers in Communication and Intelligent Systems, SCRS, India, 2021, pp. 165-176. https://doi.org/10.52458/978-81-95502-00-4-20.