Journal of **N**onlinear **A**nalysis and **O**ptimization
Vol. 15, Issue. 1, No.11: 2024
ISSN : **1906-9685**

SOFTWARE EFFORT ESTIMATION USING STACKED ENSEMBLE LEARNING TECHNIQUES

DR D Karun kumar Reddy1, M Lakshmi Nandu2, P Shyamala3, K Meghana 4,M Gayatri Vyshnavi5
1,2,3,4,5 Department of Computer Science and Engineering, Vignan's Institute of Engineering for Women, Visakhapatnam, Andhra Pradesh, India

ABSTRACT
Software Effort estimation is the process of predicting the most realistic amount of effort required to develop a software project. Accurate software effort and cost estimation are pivotal for successful project management in software development. As the application of software is extensively increased in its size and complexity, the traditional methods are not adequate to meet the requirements. To achieve the accurate estimation of software effort ensemble learning methods are being used. Addressing challenges of overestimation and underestimation, the exploration of Ensemble Learning Methods (ELMs) is undertaken to evaluate estimation accuracy. By amalgamating predictions from multiple models, ensemble learning outperforms individual ones, achieving precise estimation without reliance on a singular model. In conclusion, a comprehensive approach is presented to optimize estimation techniques, ensuring successful project outcomes within budget and schedule constraints.
Keywords: Software Effort Estimation, Ensemble Learning Techniques, Performance measurement

INTRODUCTION

In the realm of software engineering, one of the fundamental challenges lies in accurately estimating the effort required for a project's development. Software Development Effort Estimation (SDEE) serves as the compass guiding project managers and teams through the complex landscape of resource allocation, budgeting, and scheduling. At its core, SDEE is the process of forecasting the resources, manpower, time, and technologies necessary to bring a software project to fruition. This predictive exercise is not merely a procedural formality but a critical precursor to the success of any software endeavor.

Early estimation within the Software Development Life Cycle (SDLC) is paramount. It serves as a blueprint for the project team, offering clarity and direction from inception to deployment. Accurate estimations empower teams to navigate challenges effectively, ensuring timely delivery of high-quality software while adhering to budgetary constraints. Conversely, inaccurate estimations can sow seeds of discord, leading to unforeseen risks, budget overruns, schedule delays, and compromised product quality.

The stakes are high in today's competitive software industry. Companies are under relentless pressure to produce top-tier software solutions within strict timeframes and budgets. This demands a delicate equilibrium between cost and quality, where every decision carries weight and ramifications. In this dynamic environment, mastering the art of software effort estimation isn't just a best practice; it's a strategic imperative for survival and success.

LITERATURE SURVEY

In[1],The authors Suyash Shukla, Sandeep Kumar had addressed the Software Effort Estimation problem in different ways, including models developed using machine learning techniques. An experimental analysis is conducted over 17 datasets from PROMISE and ISBSG data repositories to evaluate the performance of the proposed model under different scenarios.

In[2],The author Parvas Ranjan Bal had experimented the Software Effort Estimation by using MultiLayer Perceptron (MLP).In this Desharnais dataset has been used where it is observed that R2

score of AdaBoost MLPNN is 82.213% which is highest among all the models, whereas the R2 score of MLPNN is 78.3%. In[3],The author Somya Goyal had proposed a heterogenous stacked ensemble for effective effort estimation with Artificial Neural Network (ANN) and Support Vector Regressor (SVR) as base learners. Five datasets have been used from PROMISE Repository and final result had stated that stacking is effective statistically as the proposed model improves the performance by 50.4% in MAR and 54.6% in
MMRE of base models.

EXISTING METHOD

There are several existing systems available for the software effort estimation with different techniques. Some include: SVR, KNN, MLP.

Support Vector Regression (SVR) is a powerful machine learning technique that can be applied to software
effort estimation mainly used for regression tasks. It is generally used for handling nonlinear relationships and it is used to find best hyperplane separating the effort levels based on its attributes in software effort estimation and for complex relationships. K Nearest Neighbours(KNN) is a non-parametric machine learning algorithm used for both classification and regression tasks. It is particularly used for projects with well- defined feature spaces and moderate-sized datasets can be applied to predict effort metrics based on similarities between the current project and historical data points. A multilayer perceptron (MLP) is a type of artificial neural network that consists of multiple layers of nodes (neurons) organized in a feed forward manner. It is used for classification, regression, and pattern recognition. By training the MLP on past data, it can learn patterns and relationships between various project factors and estimate the effort needed for new projects

PROPOSED METHOD

Ensemble learning has emerged as a powerful technique in the field of machine learning, offering a range of advantages over traditional individual models. This is because ensembles can effectively reduce the variance and bias of individual models, leading to better performance on both training and test data. It is preferred over other machine learning models because it combines predictions from multiple models to improve accuracy. This refers to a technique where multiple models are trained independently and then combined to make predictions. These models can be of the same type (homogeneous ensemble) or different types (heterogeneous ensemble). Examples of ensemble methods include: Random Forest, Bagging, Boosting, Stacking. Stacked ensemble, also known as stacked generalization or stacking, is a more advanced form of ensemble learning where the predictions of multiple base models are used as input features to a meta-model which then makes the final prediction. The key difference here is the introduction of a metamodel that learns to combine the predictions of the base models.

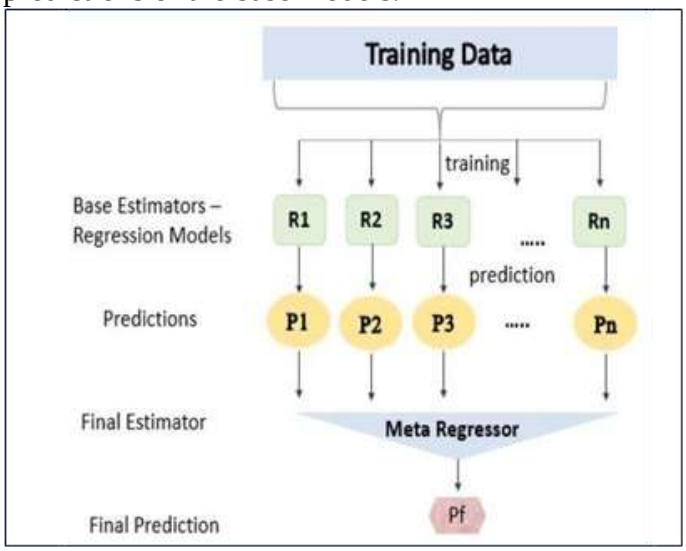

Figure I. Stacked Generalization Method – Regression.

http://doi.org/10.36893/JNAO.2024.V15I1N01.248-253

DESIGN STRUCTURE

The design structure emphasizes a systematic approach to Software effort estimation using stacked ensemble learning Techniques. Stacked ensemble learning involves combining multiple base models to improve predictive performance.

1. Data Collection and Preparation: Historical data has been collected on software development projects where datasets China has been taken. This data should include features such as project size, complexity, team experience, and other relevant factors. Then the data is cleaned by the process of normalization to handle missing values, outliers, and inconsistencies. Preprocessing the data by scaling numerical features, and performing any other necessary transformations.
2. Feature selection: In this process the features Prioritized that are highly relevant to the software development process that had a significant impact on effort estimation. Data analysis has been done to identify features that correlate strongly with the target variable (effort) and are informative for prediction. 3.Model Selection: A variety of base models that are suitable for regression task has been taken that include Random forest (RFR),Multilayer perceptron(MLP),Decision tree(DT),Bagging, Boosting. It is experimented with different algorithms to know which performs the best
4. Stacked Ensemble Aggregation: Stacked ensemble framework has been implemented where multiple models are trained and their predictions are combined using a meta-learner. The data is split into train and test sets. The predictions of the base models are taken as features to train the meta-learner and then experimented with different meta-learners.
5. Evaluation: Evaluated the performance of the stacked ensemble model using appropriate metrics such as mean absolute error (MAE), mean squared error (MSE),Mean Magnitude relativeerror(MMRE),Prediction. Compared the performance of the stacked ensemble model with individual base models to assess whether the ensemble provides a significant improvement in prediction.

CRITERIA NAME CRITERIA DEFITION

CRITERIA NAME	CRITERIA DEFITION
Mean squared error (MSE)	$\frac{1}{N} * \sum \lvert y_{true\ i} - y_{pred\ i} \rvert 2$
Magnitude of relative error (MRE)	$(-) \sum_{n}^{1} * \left\lvert \frac{(y_{true\ i} - y_{pred\ i})}{y_{true\ i}} \right\rvert$
Mean Absolute Error	$\Sigma(-) * \lvert \bar{y}_{n true y pred} \rvert$ (1)
Coefficient of Corelation (R^2)	$1 - \sum \frac{\Sigma (y_{true\ i} \bar{y}_{pred\ i})^2}{y_{true\ i}}$

Table I. Formulae of MSE, MRE, MMRE, R2

SYSTEM BLOCK DIAGRAM

http://doi.org/10.36893/JNAO.2024.V15I1N01.248-253

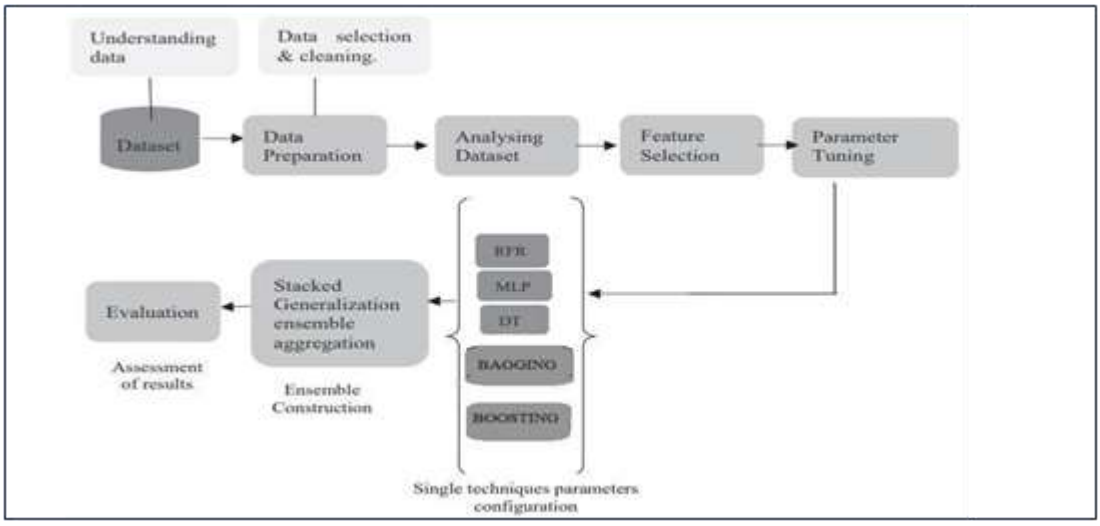

Figure II. Architecture of the proposed Experiment to construct the stacked generalization ensemble model.

RESULT ANALYSIS

Final results of the ensemble model techniques are shown in the below table:

Model	MSE	MMRE	R^2 SCORE	PREDICTION
Decision Tree	0.1406	3.1856	0.6739	67.39
Random Forest	0.11	3.9135	0.7951	79.51
Bagging	0.1078	1.7182	0.7682	76.51
GradientBoosting	0.0306	4.2092	0.8563	85.63
MLP	0.1264	4.1132	0.8963	89.63

Table II. Summarized Final Results of the developed ensemble model.

Final results of the developed stacked generalization ensemble model are shown in the below table:

Model	MSE	MMRE	R^2 SCORE	PREDICTION
Decision Tree	0.1234	1.9064	0.884	88.39
Random Forest	0.11	2.763	0.8252	82.51
Bagging	0.1077	1.5047	0.9207	92.07
GradientBoosting	0.0306	4.2092	0.7074	70.74
MLP	0.1264	2.374	0.9163	91.63

Table III. Summarized Final Results of the developed stacked ensemble model.

Results of Statistical Tests:

As we discussed earlier the critical value for k=3, n=10 and alpha/p-value = 0.05 which is 7.8. If Friedman statistic is greater than critical value, we reject the null hypothesis else we accept the null hypothesis:

Dataset	Friedman ChiSquare Statistic	P-Value
China	12.96	0.0114

Table IV. Results of Friedman ChiSquare Test

http://doi.org/10.36893/JNAO.2024.V15I1N01.248-253

Results after feature selection performed and software effort estimation is predicted:

Figure III. Software effort is predicted

Graphical representation of results before and after implementing feature selection, stacked ensemble generalisation are shown below:

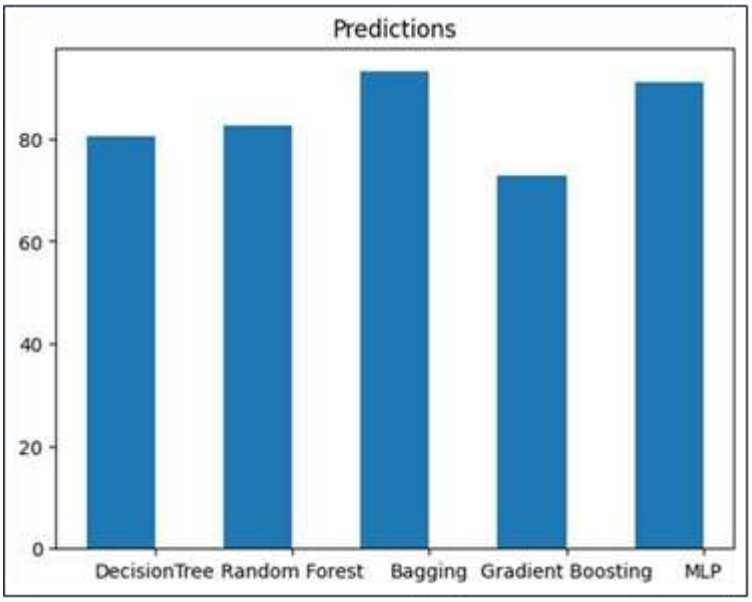

Figure IV Graphical representation of results stacked ensemble generalisation

CONCLUSION

Estimating Software Development Effort (SDE) is crucial for project managers, aided by SDEE models acting as decision support systems. Flexibility and robustness are key concerns for any estimation model, especially across diverse data. The choice and performance of machine learning algorithms depend on the dataset, as no single algorithm fits all problems. Combining strengths of different techniques can enhance accuracy in SDE estimation. This thesis focuses on developing an effective SDE estimation model. Stacked generalization ensemble method is chosen after thorough research. Evaluation compares its efficiency with existing models.

REFERENCES

[1] Self-Adaptive Ensemble-based Approach for Software Effort Estimation" by Suyash Shukla, Sandeep Kumar [2023]

[2] Analysing Effect of Ensemble Models on Multi-Layer Perceptron Network for Software Effort Estimation" by Pravas Ranjan Bal [2019]
[3] "Improving Estimation Accuracy Prediction of Software Development Effort: A Proposed Ensemble Model" by Yasir Mahmood, Nazri Kama, Azri Azmi [2021]
[4] On the Interpretability of a Tree-based Ensemble for Predicting Software Effort" by Assia Najm, Abdelali Zakran[2023]
[5]A Systematic Literature Review of Machine Learning Techniques for Software Effort Estimation Models" by Pooja Brar, Dr.Deepak Nandal
[6] Comparing Stacking Ensemble and Deep Learning for Software Project Effort Estimation" by Huynh Thai hoc, Radek silhavy , Zdenkya Prokopova , and petr silhavy[2023]
[7] Ensemble of Learning Project Productivity in Software Effort Based on Use Case Points" by Mohammad
Azzeh, Ali Bou Nassif, Shadi Banitaan, Cuauhtémoc López-Martín[2018]
[8] "Software Development Effort Estimation Techniques Using Long Short Term Memory" by Farah B.Ahmad, Laheeb M. Ibrahim[2022]
[9] "Effective Software Effort Estimation using Heterogenous Stacked Ensemble" by Somya Goyal [2022]

http://doi.org/10.36893/JNAO.2024.V15I1N01.248-253