African Journal of Pure & Applied Sciences

ISSN: 2710-0758
DOI: https://doi.org/10.33886/ajpas.v5i2.565

Journal homepage: http://journal.ku.ac.ke/index.php/AJPAS

Research Article Open Access

Software Cost Estimation through Stacking Ensemble Approach: An Evaluation Study

Ophori Ufuoma Anita[1] & Egbokhare Francisca[2]

Department of Computer Science,
University of Benin, Benin City, Nigeria
ufuoma.ophori@uniben.edu[1] fegbokhare@uniben.edu[2]

Corresponding email: ufuoma.ophori@uniben.edu

Received 30th June, 2024; Accepted 28th October 2024; Published online; 20th December 2024

How to Cite:
Ophori , A. . U., & Egbokhare , F. Software Cost Estimation through Stacking Ensemble Approach: An Evaluation Study. *African Journal of Pure and Applied Sciences, 5(2),* 20–28. https://doi.org/10.33886/ajpas.v5i2.565

ABSTRACT

Accurately estimating software development costs is paramount for software project success. Global attention has focused on addressing inaccuracies in software cost estimating models leading to numerous approaches. Although machine learning algorithm techniques have witnessed improvement, the challenge of varying bias and variance from weaker learners persists, leading to the adoption of ensemble learning techniques for increased accuracy. This paper extends previous work on ensemble learning Software Cost Estimation (SCE) approach through the use of a Stacking Ensemble Learning Model for SCE (SELM-SCE) to improve the accuracy of software cost estimation. A combination of unsupervised and supervised machine learning methods were used on software projects. Two datasets (Desharnais and Maxwell) for software cost estimation were used for the model training and validation using cross-validation techniques. The proposed SELM-SCE model developed was subjected to two training models: K-Means and SMOTE technique to solve the problem of overfitting and poor performance caused by data imbalance. The performance of the proposed SELM-SCE model was first evaluated against three regression performance metrics: coefficient of determination (R^2), root mean squared error (RMSE), and mean absolute error (MAE). The results showed that the SMOTE SELM-SCE without unsupervised machine learning improves the accuracy of the model performance. The performance of SELM-SCE was compared with existing models using prediction accuracy, precision, recall, and F1-score classification performance metrics. The results from the SELM-SCE model showed remarkable improvement when compared to the existing model.

Keywords: Software Cost estimation, Stacking Ensemble, Machine learning, Project Cost Prediction

INTRODUCTION

Software cost estimation (SCE) is the method of estimating the cost and effort required to complete a software system. Efficient SCE is a major concern in software project management (Azzeh *et al* 2018). Research on algorithmic cost estimation and their performances was reported to be very low with poor predictive accuracy(Jorgensen & Shepperd, 2007) due to poor sizing inputs and inaccurate cost driver ratings. This limitation led to the use of hybrid methods Thus, soft computing techniques have been adopted to enhance the prediction accuracy of SCE (Venkataiah *et al.*, 2017). Despite the performance, soft computing models were impeded by high biases and variance, which lowered the accuracy of SCE arising from the inability to address certain data complexities. This inherent weakness led to the development of ensemble methods which have been well adopted to produce viable, robust, and more accurate SCE models (Kumar & Datta, 2015; Hidmi *et al.*, 2017).

The ensemble method aggregates multiple predictions from several models to obtain a composite model that

outperforms each model (Lofstrom, 2015). The strength of ensemble learning methods is the identification and combination of weaker models to obtain more robust and accurate predictions. Several ensemble methods have been adopted for SCE (Braga *et al.*, 2007; Kultur *et al.*, 2008; Nassif, 2012; Minku & Yao, 2013; Elish, 2013; Shin 2015; Hidmi & Sakar 2017; Pospieszny *et al.*, 2018; Alhazmi & Khan 2020, Carvalho *et al.* 2020, Varshini *et al.* 2021 and Kumar *et al.*, 2023). Most existing ensemble methods focused more on bagging and boosting approaches. The notion of stacking ensemble has rarely been explored for SCE. Also, Kalia (2018) and EL Aissaoui *et al.* (2019) suggested that researchers should investigate the combination of unsupervised and supervised Machine Learning (ML) algorithms as base learners in the design of ensemble models. Though most existing ensemble models can accurately predict with higher accuracy compared to supervised ML models, this study proposed a stacking ensemble technique using unsupervised ML algorithms stacked upon supervised ML algorithms tagged SELM-SCE model. This study developed a Stacking Ensemble Learning Model for Software Cost Estimation (SELM-SCE) to enhance prediction accuracy. Synthetic Minority Over-Sampling Technique (SMOTE) was applied to create supplementary synthetic datasets to further enhanced the prediction accuracy. Finally, the results were compared with Hidmi & Sakar (2017) boosting ensemble model.

MATERIALS AND METHODOLOGY

The SELM-SCE model development design is shown in Figure 1. The model used the default stack generalization technique to utilize the proposed SELM-SCE model but was slightly modified. The proposed model stacked an unsupervised machine learning algorithm (base learner) on a supervised ML algorithm, using a variety of learning algorithms however is more effective than using techniques that with similar algorithms and output (Kalia, 2018), unlike conventional staking methods that use only supervised machine learning models The model was constructed using two main blocks: the data preprocessing and stack ensemble block. The data preprocessing block involved data collection and preprocessing steps while the stack ensemble block carried out the actual stacking ensemble training. This block ensured that accuracy enhancement was achieved through the application of supervised and unsupervised machine learning methods. It exploited the advantages of both machine learning algorithms. These algorithms include the stacking, K-Means, K-Nearest Neighbor, and LASSO Regression algorithms. The stacking phase receives the training and test data from the knowledgebase and stacks both the supervised and unsupervised learning algorithms to achieve the process prediction enhancement. The supervised algorithms used in this phase include the K-Nearest Neighbor (KNN) and LASSO regression algorithms while the unsupervised ones include the K-Means. The K-Means are stacked upon the supervised learning algorithms (KNN and LASSO regression Algori

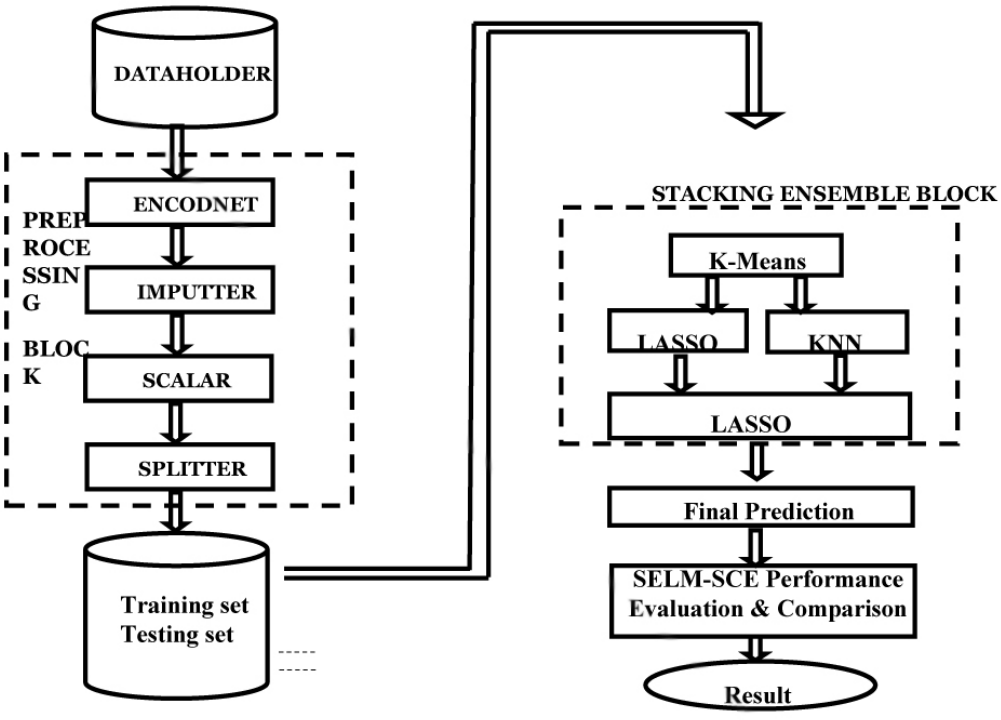


Figure 1: Proposed Stacking Ensemble Learning Model for Software Cost Estimation (SELM-SCE Model)

Dataset Description

In this study, data preprocessing steps of one-hot encoding, multiple imputations by chained equations, standardization, and holdout validation were conducted on two publicly available datasets: Desharnais (63 data points) and Maxwell (81 data points) collected from a standardized repository:Predictor Model in Software Engineering Software Repository http://promise.site.uottawa.ca/SERepository/dataset/desharnaise.arff.), which are open to researcher. The first step involved converting categorical features into numerical representations through one-hot encoding, ensuring compatibility with the planned regression models. This addressed five features in the Desharnais dataset and three in the Maxwell dataset. Next, the three missing software feature values encountered within the Maxwell dataset were treated using Multiple Imputation by Chained Equations (MICE). This imputation technique, utilizing linear regression as its core model, effectively handled the missing data while preserving valuable information and preventing the exclusion of potentially useful features.

Data imputation of both datasets was subjected to holdout cross-validation for evaluation and generalizability assessment. This involved randomly splitting each dataset into training and testing sets using a 7:3 ratio. Finally, all software features except the target variable (*Effort*) underwent standardization via scaling. This ensures features have comparable scales, promoting model convergence and interpretability. The choice of one-hot encoding for categorical features aligns with established practices in regression modeling. MICE imputation, specifically employing linear regression within its methodology, stands out as a practical solution to address missing values while conserving information and avoiding the discarding of features that could significantly impact software cost prediction. Holdout validation represents a commonly adopted technique for model evaluation and generalizability assessment in the field. Lastly, standardization serves to improve model convergence and interpretability by ensuring features possess similar scales. The final preprocessing step involved standardization, aiming to bring all software features (except the target variable) onto a common scale. This improves model convergence, and interpretability, and reduces the influence of features with larger scales on the model's predictions. Standardization employed the Z-score transformation**.** (1) defines the standardization equation (Bhandari, 2020).

$$Z = \frac{\chi - \mu}{\sigma} \quad (1)$$

Where Z = standard score, χ =observe software project feature values, μ= Mean of the software project features value, and σ = standard deviation

For the SELM-SCE model, 70% of the data set was used for training and the remaining 30% of the data set was used for testing.

To mitigate the negative impact of using a limited number of data points on the performance accuracy of our SELM-SCE model, the SMOTE is introduced on the dataset to overcome the problem associated with imbalanced datasets such as over-fitting and poor performance. The decision to employ SMOTE was based on the understanding that a sufficient number of records in the dataset is crucial for obtaining satisfactory results. Table 1 shows the summary of both datasets after applying SMOTE to the whole dataset.

Table 1: Summary of both Datasets (SMOTE Datasets)

Dataset	Before SMOTE	After SMOTE	SMOTE Training set	SMOTE Test Set
Desharnais	81	138	110	28
Maxwell	63	110	88	22

Modelling

The selected ML algorithms were trained by calling the SELM-SCE model and the training comprises three steps. The first step involves clustering of the training dataset where 70% of the training data was passed through the k-means algorithm to generate two clusters.

The second step involves the training of two base learners, Lasso regression and KNN algorithms. The respective trained models were used to estimate the test dataset.

In the third step called the meta-learner, the Lasso regression learning algorithm was used to represent the last training. The output estimates of Lasso regression and KNN of step 2 were selected, combined, and fed into the meta-learner as the new training input that trained the lasso regression (meta-learner ML algorithm). The final estimate on the original test data was performed using the meta-learner. The experiment was set up by constructing four different stacked ensemble models to answer a research question, i.e., "if an unsupervised machine learning algorithm using K-Means in stacking ensemble improve software cost estimations?" The following four models were experimented in the research:

- SELM-SCE 1 - implementing without clustering
- SELM-SCE II - implementing with clustering
- SELM-SCE III - implementing without clustering (SMOTE)
- SELM-SCE IV - implementing with clustering (SMOTE)

a. Training Model without Clustering: The experiments in this section used both LASSO regression and KNN algorithms. The model consists of two levels, the first level comprised of LASSO and KNN, whilst the second comprised of LASSO as the meta-learner. These ML algorithms were trained on the 70% train set and validated using the test set. The performance of Lasso and KNN was highly influenced by the parameters used which are the alpha and n-neighbor parameters, to adjust the sensitivity of the results achieved.

b. Training SELM-SCE with Clustering: In this section, the heuristic method was used to determine the number of clusters and validated using a standardized statistical measure referred to as the elbow method (Lopez-Rubio et al., 2018) and obtained k as two (2) as the number of clusters used. However, a range of k values from 2 to 4 was experimented with and depicted in Figures 2 to Figure 3 proving the justification of our approach applied using both the original (Non-SMOTE) and the SMOTE datasets. The adequacy of the heuristic adopted in the study is shown in Figure 3. Algorithm 1 shows the algorithm for the elbow method.

- **Input:** training data x= {maxData, desData}
- **Output:** optimal_k
  //k: = the optimal number of clusters
- initialize an empty list to store the software cost values for k
  Software cost_Values =[]
  clusters=KMeans(data x, k)
  softwarecost=caluculatesoftwareCost(data x, cluster)
  plot the software cost values against the number of clusters (k)
  find the "elbow point"(softwarecost_values)
  return optimal _k

Algorithm 1: Elbow Method

Comparison Phase

To ensure that the aim of the study is achieved, the SELM-SCE models were evaluated as a classification model to demonstrate the comparison with boosting ensemble techniques (Hidmi & Sakar., 2017) by converting the continuous scores of estimated software cost to categorical scores. Table 2 presents the range of scores used as a class after discretization.

Table 2: Discretization (NON-SMOTE and SMOTE DATASET

Class	Dataset		
	Desharnais Dataset		
	Actual Software cost	Non-SMOTE Data (81 Cases)	SMOTE Data (134 Cases)
1	0-3500	38	59
2	>3500	43	79
	Maxwell dataset		
		No. of Cases(63)	Total Cases: 110
1	0-5000	31	47
2	>5000	32	63

Performance Evaluation Metrics

A: Regression metrics: The coefficient of determination (R^2), root mean square error (RMSE) and mean absolute error (MAE) were adopted as the regression matrices to evaluate the SELM-SCE performance.

1. The coefficient of Determination (R^2) also called R-square defines model accuracy. (Chicco *et al.*, 2018). In this study, it is the square of the correlation between the SELM- SCE model predicted and actual values in a given dataset as presented in Equation 2.

$$R^2 = 1 - \frac{\text{unexplained variation}}{\text{total variation}} = 1 - \frac{\sum_{i=1}^{m}(xi-yi)^2}{\sum_{i=1}^{m}(\bar{y}i-yi)^2} \quad (2)$$

where $\sum(xi\text{-}yi)^2$ is the Sum of squared regression and $\sum(yi\text{-}yi)^{2\,i}$ is the sum of the square total

2. The root mean square error (RMSE) measures model performance in terms of errors (Chai et al., 2013), It is the square root of the differences between actual and SELM-SCE predicted values taking the mean across the sample. Then, the square root of this mean is taken to arrive at the final value as presented in Equation 3

$$RMSE = \sqrt{\sum_{i=1}^{n}\frac{(\hat{y}i - yi)^2}{N}} \quad (3)$$

where yi is the actual software cost and $\hat{y}i$ is SELM-SCE predicted software cost.

3. Mean Absolute Error (MAE) helps evaluate the average error size within a group of predictions. It helps to verify the data point for the differences between SELM-SCE predictions and the related actual values (Frías-Paredes *et al.*, 2018) as shown in Equation 4.

4.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |xi - x| \quad (4)$$

Where n is the number of samples or instances and Σ|xi – x| = sum of absolute errors

B: Classification Metrics: Beyond the accuracy measure used by the reference model, we further added precision, recall, and F1-score measures from the confusion matrix to extend this research finding. The formula for computing accuracy, precision, recall, and F1 score are shown in Equations 5, 6, 7, and 8,

i. Accuracy (%): This shows the fraction of prediction the proposed SELM-SCE model got right. It's the number of correct predictions divided by the total number of predictions multiplied by 100

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} * 100 \quad (5)$$

ii. Precision (%): it is used to calculate the SELM-SCE model's ability to classify positive values correctly. it is the ratio of correctly predicted positive observations. High precision indicates a low false positive rate while low precision indicates a large number of false positive

$$\text{Precision} = \frac{TP}{TP+FP} * 100 \quad (6)$$

iii. Recall (%): Measures how likely our model, SELM-SCE will predict correctly the software effort of all software projects when it is present and exhibited. It is the ratio of true positive (TP) to the sum of true positive (TP) and false negative (FN)

iv.

$$Recall = \frac{TP}{TP+FN} * 100 \quad (7)$$

v. FI - score: this is a metric combining both precision and recall. (it is the harmonic mean of recall and precision. It takes both false positives and false negatives and helps in the place of severe class imbalance in the Maxwell and Desharnais dataset used (from the SELM-SCE model). This is useful when you need to take both precision and recall into account.

$$F1-\text{Score} = \frac{2}{\frac{1}{Recall}+\frac{1}{Precision}} * 100 \quad (8)$$

Where, True positive (TP) = predicted value matches the actual value (Actual positive, SELM-SCE predicted positive),

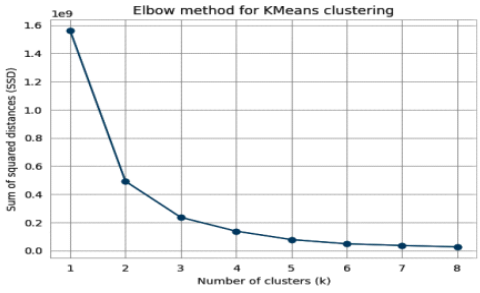
True negative (TN) = the predicted value matches the actual value (Actual negative, SELM-SCE predicted negative value)

False positive (FP) = the predicted value was falsely predicted (Actual positive, SELM-SCE predicted Negative value)

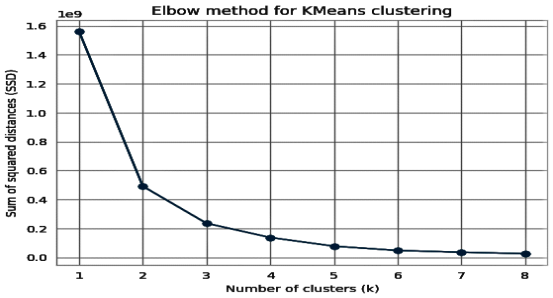
False negative (FN) = predicted value was falsely predicted (Actual negative, SELM-SCE predicted positive value).

IMPLEMENTATION AND RESULTS

The models under discussion were implemented using Python programming language. Python libraries such as Scikit-learn form the heart of model development for KNN and Lasso Regression, and Matplotlib and Seaborn packages for plotting graphs/charts. The hardware used was an Intel Core i7 with 12 GB RAM and 500GB hard disk. The experiment was conducted using two datasets-Desharnais and Maxwell. Both have imbalanced datasets (a case of having to oversample). The Synthetic Minority Oversampling Technique (SMOTE) was applied to address imbalance or oversampling. Figure 2 shows the elbow method on the non-SMOTE set. Figure 3 shows the elbow method after SMOTE was applied.

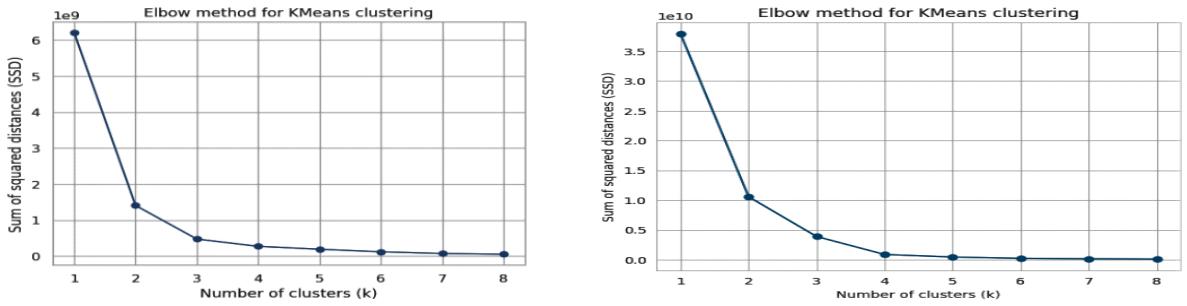


a. Desharnais Set



b. Maxwell set

Figure 2.: Elbow Methods Plot (Non-SMOTE Dataset)



a. Desharnais Set b. Maxwell set

Figure 3: Elbow Method Plot (SMOTE Applied Dataset)

In summary, table 3. shows the dataset clustering output derived from both datasets.

Table 3: Training clustering set summary

Training Dataset	Cluster A	Cluster B
Desharnais(81)	68	13
Maxwell (63)	59	4
Desharnais SMOTE (138)	73	65
SMOTE Maxwell (110)	29	81

Stacked Ensemble Method Implementation

The experimental process is described as follows:

i. Training LASSO and KNN using the training set,

ii. The results obtained from (1) serve as input for the meta-leaner (LASSO), and

iii. The selection of the optimal parameters as described above was obtained using R^2, RMSE, and MAE metrics.

Comparison of models

Table 4: Summary of Performance Analysis for ALL SELM-SCE Models without SMOTE Technique

Model Performance on Datasets			
Desharnais Dataset			
	R^2 (%)	**RMSE**	**MAE**
SELM-SCE I (without KMeans)	73.73	3543.65	2410.45
SELM-SCE II (with KMeans)	59.12	3281.00	2439.06
Maxwell Dataset			
SELM-SCE I (without KMeans)	77.36	2903.49	2299.13
SELM-SCE II (with KMeans)	71.36	3265.92	2548.28

Table 4 shows the performance analysis for the first two (2) models without SMOTE technique. In both datasets, the SELM-SCE I (without K-Means) obtained a higher R^2 value (73.73 and 77.36) and also showed consistent improvement in R^2, RMSE, and MAE performance values in the Maxwell dataset. A smaller RMSE indicates an improvement in model performance

Table 5: Summary of Comparative Performance Analysis of all SELM-SCE Models with SMOTE Technique

Model Performance on Datasets			
Desharnais Dataset			
	R^2 (%)	**RMSE**	**MAE**
SELM-SCE III (SMOTE without KMeans)	**93.15**	**1687.08**	**1188.00**
SELM-SCE IV (SMOTE with KMeans)	76.33	3134.55	2103.41
Maxwell Dataset			
SELM-SCE III (SMOTE without KMeans)	**94.67**	**4532.79**	2550.91
SELM-SCE IV (SMOTE with KMeans)	92.91	5139.83	**2543.70**

In Table 5, SELM-SCE III (with K-Means) for both datasets obtained the following R^2 values of 93.15% and 94.67% while SELM-SCE-IV obtained 76.33% and 92.91% respectively. This also showed that the SELM-SCE III (with K-Means) model improves the accuracy of the model performance.

When the research question was considered "Can the accuracy of software cost estimation be enhanced if an unsupervised learning model is stacked with supervised learning models"? It is observed that the proposed SELM-SCE model without k-means generally improves the prediction of software cost. However, the report that an unsupervised algorithm in our stacked ensemble improves model performance could not be established. This could be a result of the presence of small samples observed in some of the cluster sets when training and this was considered a threat to validating the effect of an unsupervised machine learning algorithm in our proposed SELM-SCE with K-Means model.

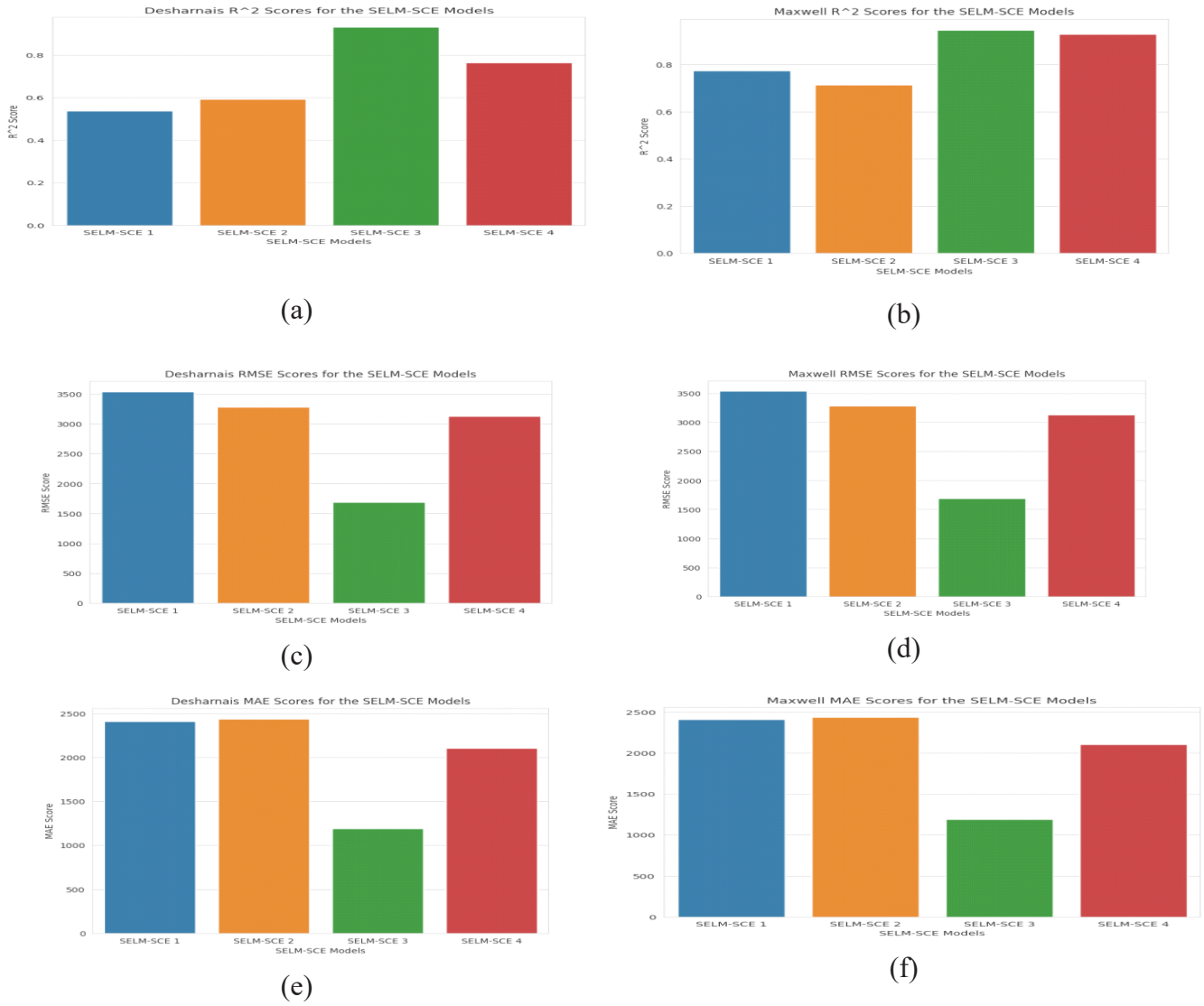


Figure 4(a-f): Comparison of Predictions of SELM-SCE Models based on Variance and Error Rate

Figure 4 (a-f) presents a bar chart of R^2, RMSE, and MAE performance metrics indicating the SELM-SCE model that well predicts software cost with the highest variation of prediction and lower error rate. The y-axis represents the percentage of the performance measurement while the x-axis shows the individual SELM-SCE models.

Comparing our Model with an Existing Boosting Ensemble

Table 6: Summary of the classification metrics results of all models using the Desharnais dataset

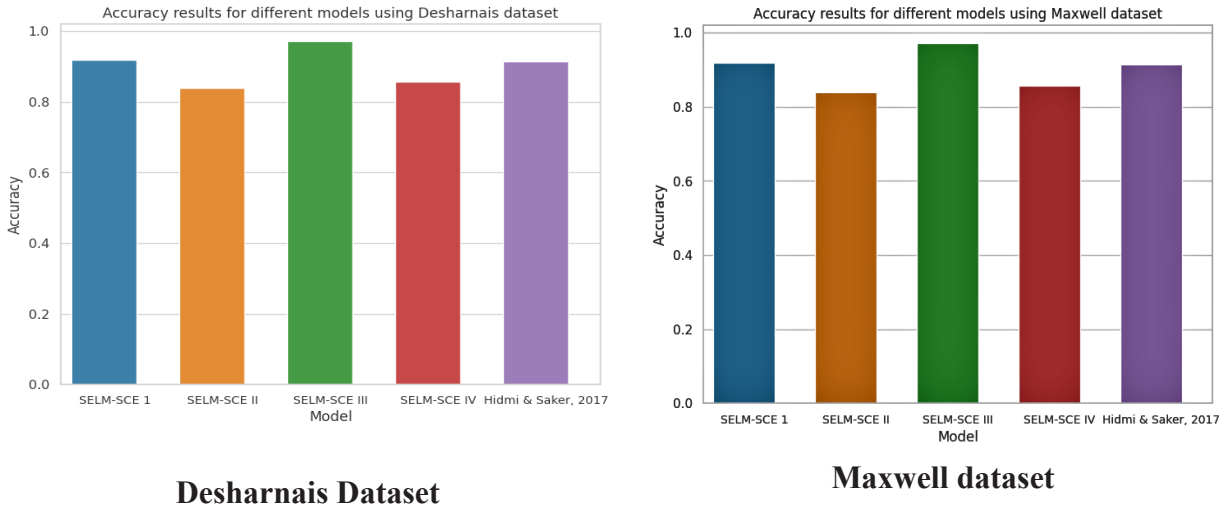
Model	Desharnais Dataset				
	Accuracy (%)	% Accuracy Improvement	Precision (%)	Recall (%)	FI Score (%)
Hidmi & Saker (2017)	**91.35**				
SELM-SCE 1 (without K-Means)	92.0	0.65%	72.72	92.30	81.25
SELM-SCE II (with K-Means)	84.00	Less 7.35	81.25	92.85	86.66
SELM-SCE III (SMOTE without K-Means)	**97.14**	**5.79%**	**92.85**	**100**	**96.29**
SELM-SCE IV (SMOTE with K-Means)	85.71	Less 5.64%.	71.42	**90.90**	**8.0**

Comparatively, our proposed SELM-SCE model with the existing boosting ensemble model by Hidmi & Sakar (2017) was evaluated based on the accuracy performance metric. In the Desharnais dataset, it was observed in Table 6 that the SELM-SCE 1 (without K-Means) model performed better than the existing model and was able to improve the accuracy by 0.65% while SELM-SCE 11(with K-Means) model achieved a 7.35% reduction in improvement. Similarly, it was observed that the stacked ensemble with the SMOTE technique, SELM-SCE 111 without the K-Means model performed better than the existing boosting model with an accuracy of 5.79% improvements while the SELM-SCE IV (with K-Means) model did not improve the accuracy by 5.64%.

Table 7: Summary of the classification metrics results of all models using the Maxwell dataset

Model	Maxwell				
	Accuracy (%)	% Accuracy Improvement	Precision (%)	Recall (%)	FI Score (%)
Hidmi & Saker (2017)	**85.48**				
SELM-SCE 1 (without K-Means)	89.47	3.99%	92.85	92.85	92.85
SELM-SCE II (with KMeans)	84.21	Less 1.27%	92.85	0.86.66	89.65
SELM-SCE III (SMOTE without K-Means)	**96.96**	**11.48%**	**92.85**	**100**	**96.29**
SELM-SCE IV (SMOTE with K-Means)	90.90	5.42%	78.57	100	88.0

In the Maxwell dataset, it was observed in Table 7 that our experiment showed that the use of proposed SELM-SCE 1 (without K-Mean) and SELM-SCE 111 (SMOTE without K-Means) significantly increases the performance of SCE except SELM-SCE 11(with K-Mean) and SELM-SCE IV (SMOTE with K-Mean) model that shows that the models fall below the performance accuracy of the existing boosting ensemble model. In both datasets, SELM-SCE III (SMOTE without K-Means) consistently demonstrated higher classification accuracy values compared with the existing boosting ensemble models, indicating its effectiveness in improving predictive accuracy of software cost by the 5.79% and 11.48% improvement as shown in Table 6 and 7 respectively.



Desharnais Dataset **Maxwell dataset**

Figure 5: Comparison of Predictions of SELM-SCE Models with Existing Boosting Ensemble (Hidmi & Sakar., 2017) based on Accuracy

In Figure 5, we compared the proposed model with the existing method using a bar chart. the bar chart portrays the improvement of our proposed SELM-SCE models based on the accuracy performance metric

CONCLUSION

At the end of this study, our analysis has shown that the SELM-SCE model can be as effective as other widely used software cost estimation models and recommended that the proposed SELM-SCE model should be used as an accurate SCE model in real-world applications to assist software estimator in predicting software cost and in applying SMOTE technique with both supervised and unsupervised ML algorithms in the stacking ensemble model for software cost estimation, further investigation should be carried for further improvement using other combination techniques to determine the best. The experiment was done to evaluate K-nearest neighbor, K-means, and Least Absolute Shrinkage Selection Operator (LASSO) by applying them to form different stacking ensemble techniques and applying them in two different datasets (Desharnias and Maxwell datasets) to make effort prediction for a new software development project. The results obtained show that the proposed model accuracies were 96.96% and 97.14% respectively, and performed better than the existing work carried out by Hidmi & Sakar (2017) who obtained an accuracy of 91.35% using the Boosting ensemble model. One major distinctive work done in this study includes the introduction of SMOTE techniques which further improved our proposed model by addressing the problems associated with overfitting showing the importance of the use of large datasets for learning. The successful application of the Stacking Ensemble Learning Model for Software Cost Estimation (SELM-SCE) has confirmed the predictive capability of the model. This achievement stands to greatly benefit other researchers seeking to enhance their predictive models and advance their work in this field. Finally, when deployed in a real-life setting, it will help and encourage project estimators and managers alike to develop and deploy high-quality software projects on time and within budget. For future work, other machine learning techniques can be used for both classification and regression problems and other datasets can also be used for more experiments and training of the models like Bagging, voting ensemble, and their variant models to further improve the existing works. In addition, the need to use other SMOTE techniques cannot be overemphasized as seen in this study, and may also help to further improve the model.

REFERENCES

Azzeh, M., Nassif, A. & Banitaan, S. (2018). Comparative analysis of soft computing techniques for predicting software effort-based use case points, *IET Software*, 12 (1), 19–29.

Bhandari, A. (2020). *Feature Scaling for Machine Learning: Understanding the Difference Between Normalization vs. Standardization*. Retrieved August 23, 2023, from https://medium.com/analytics-vidhya/feature scaling-for-machine-learning-normalization-vs-standardization-34daa2d4f707

El Aissaoui O., El Alami Y., Oughdira L. & EL ALLIO Y. (2019). Combining supervised and unsupervised machine learning algorithms to predict the learners' learning styles. *Procedia Computer Science*, 148, 87–96.

Frías-Paredes, L., Mallor, F., Gastón-Romeo, M. & León, T. (2018). Dynamic mean absolute error, a new measure for assessing forecasting errors. *Energy Conversion and Management*,162, 176–88.

Hidmi, O. & Sakar, B. (2017). A Software Development Effort Estimation Using Ensemble Machine Learning. *International Journal of Computing, Communications and Instrumentation Engineering*, 4 (1), 143-147.

Jorgensen, M. & Shepperd, M. (2007). A systematic review of software development cost estimation studies. *IEEE Transactions on Software Engineering*, 33 (1), 33–53.

Kalia, P. (2018). Stacking Supervised and Unsupervised Learning Models for Better Performance. *International Research Journal of Engineering and Technology*, 5(09), 334-338

Kumar, B.K., Bilgaiyan, S. & Mishra, B.S.P. (2023). Software Effort Estimation through Ensembling of Base Models in Machine Learning using a Voting Estimator. *International Journal of Advanced Computer Science and Applications*, 14(2), 172-181.

Kumar, A. & Datta, U. (2015). An Effort Estimation Model for Software Development using Ensemble Learning. *International Journal of Computer Applications*, 115(21), 37-41.

Löfström, T. (2015). On effectively creating ensembles of classifiers: Studies on creation strategies, diversity and predicting with confidence. (Publication No. 15-009) [Doctoral dissertation, Department of Computer and Systems Sciences, Stockholm University].

López-Rubio, E. A., Palomo, E. J. & Ortega-Zamorano, F. (2018). Unsupervised learning by cluster quality optimization. *Information Sciences*, 436–437, 31-55.

Shin, Y. (2015). Application of Boosting Regression Trees to Preliminary Cost Estimation in Building Construction Projects. *Computational Intelligence and Neuroscience,* 2015(4)

Venkataiah, V., Mohanty, R., & Nagaratna, M. (2017). Review on Intelligent and SoftComputing Techniques to Predict Software Cost Estimation. *International Journal of Applied Engineering Research,* 12(22), 12665 -12681.