Innovations in Systems and Software Engineering
https://doi.org/10.1007/s11334-024-00581-2

ORIGINAL ARTICLE

SENSE: software effort estimation using novel stacking ensemble learning

Anupama Kaushik[1] · Kavita Sheoran[2] · Ritvik Kapur[1] · Nikhil Bhutani[2] · Bhavesh Singh[2] · Harsh Sharma[1]

Received: 29 May 2023 / Accepted: 28 August 2024

Abstract
The volatile factors involved in software cost estimation have long been an occlusion for the software development life cycle. The inaccuracy they lead to during the estimation process has had an implacable effect on the stakeholders concerned. This can be mitigated by using machine learning algorithms to estimate the cost, which significantly reduces the volatility of the process and has more reliable results. Thus, implementing stacking on various datasets with SVR, LightGBM, K-nearest neighbours and Random Forest in level-0 and Ridge Regression in level-1 has given highly accurate results. The SENSE-Software Effort Estimation using Novel Stacking and Ensemble learning- model proposed in this study is substantiated on six datasets China, Kemerer, Albrecht, Nasa93, ISBSG and Maxwell and evaluated using MAE, RMSE, R^2, PRED and MMRE as evaluation metrics. We find that the proposed model displays competent performance in experimental evaluation and statistical analysis in comparison to the other studies used in the work.

Keywords Software cost estimation · Ensemble learning · Stacked generalisation · Bayesian optimization · Random forest · LightGBM · K-nearest neighbours

1 Introduction

Software cost estimation is an effective technique to compile all the expenses related to software development.

Without a precise cost estimate, the resources needed will either be underestimated, which will require more time, involve fewer developers, and result in inadequate training and study, or they will be overestimated, which will be expensive for the customer and wasteful for the software company. However, the failure of the project will occur in either situation. Software cost planning is, therefore, one of the most critical phases of software development. Due to numerous factors that must be considered, it is a challenging process [1], leading to multiple models and approaches for software cost estimation [2].

✉ Anupama Kaushik
anupama@msit.in

✉ Kavita Sheoran
kavita.sheoran@msit.in

[1] Department of IT, Maharaja Surajmal Institute of Technology, New Delhi, India

[2] Department of Computer Science, Maharaja Surajmal Institute of Technology, New Delhi, India

Software cost estimation is challenging due to many reasons: highly changing nature of software due to changes in requirements; lack of historical data which can serve as the basis of estimation; dependencies on new technologies and platforms making software complex; lack of standardization in cost estimation techniques; and reliance on human factors leading to biases and errors. Thus, by handling complicated factors, lowering human bias, saving time, and enhancing accuracy through data-driven predictions, machine learning improves software cost estimation. In order to facilitate more informed decision-making and improved project management, it provides scalability, predictive analytics, real-time updates, and configurable models. It integrates smoothly with contemporary technologies.

In order to overcome these challenges researchers are continuously working and enhancing their estimation techniques which can provide more accurate estimations.

So, the objective of the current study is to find a software cost estimation model with maximum efficiency and accuracy. The suggested solution is through the proposed technique "SENSE". This technique uses stacking as its base and it combines the strength of the techniques random forest, K-nearest neighbors, SVR, Light GBM and ridge regression. Due to variety of techniques the stacking model is able to cap-

Published online: 14 September 2024

ture the diversity existing in software project data. It enhances robustness by lowering the impact of weaknesses of various models individually. Stacking also enables to handle the complex relationships and dependencies present in the project data which is difficult for an individual model.

Initially software cost estimation started with algorithmic models which comprises of COCOMO [3], SLIM [4], SEER-SEM [5], Function Point [6] and COSMIC [7]. But these models did not stand for a longer time due to various limitations [8]. These models were not successful as they required definite inputs which was difficult to receive during initial stages of software development. They were unable to handle the categorical inputs and were deficit in reasoning capabilities. So, non-algorithmic models took over based on soft computing techniques comprising of Fuzzy Logic, Artificial Neural Networks, Evolutionary Computation and many more. A vast amount of work was done in this direction [9–12].

Now-a-days researchers are moving towards ensembling models as they have their own unique strength by combining multiple models. All the techniques given for software cost estimation in the past have their own strengths and weaknesses. There is no standard technique present, so the search for new technique continues for more accurate predictions.

This paper is organized as follows: In Sect. 2 a brief description of related work is given which motivated the framework of this study. Section 3 highlights and describes all the background techniques used in this study, followed by Dataset Description and preprocessing in Sect. 4. Proposed Work is explained in detail in Sect. 5 with all the steps involved in the SENSE framework. In Sect. 6, the Evaluation Criteria which are used to compare and analyze the accuracy of the SENSE model are described. In Sect. 7 Experimental Evaluation is presented which provides tabular and graphical representation of the results, followed by Statistical Analysis in Sect. 8. Threats to validity are discussed in Sect. 9. The study is finally concluded in Sect. 10.

2 Related work

For the goal of estimating software costs, there has been a lot of prior research, and various machine learning models have also been put forward. The SABE model, proposed by Kaushik et al. [13], puts forward an analogy-based estimation (ABE) method using stacking, which is a solution function aimed to ameliorate analogy-based estimation prediction. It is evaluated on standard accuracy (SA) and median magnitude of relative error (MdMRE) among other evaluation metrics. In Random Forest-Based Stacked Ensemble Approach by Priya Varshini et al. [14], the first-level classifiers considered were, Random Forest (RF), decision tree, Lasso and Elastic-Net Regularized Generalized Linear Models (Glmnet), and SVMRadial. Second level classifier used was Random Forest and resulted in proposed stacking model providing better results than single models. Sampath Kumar and Venkatesan [15] utilised the Base Learners-Neural Network, Linear Regression, Random Forest Regression and for meta learners the Support Vector Regression was used. Using PRED and the mean magnitude of relative error (MMRE) they observed an improved error score. Sakhrawi et al. [16] used projects based on the scrum framework for their publication upon which the first model uses three ML techniques-Random Forest Regressor (RFR), LinearSVR, and Decision Tree Regressor (DTRegr), and the second model uses StackingRegressor. This study achieved the following results- Mean Square Error (MSE) = 0.406, MAE (Mean Absolute Error) = 0.206, and Root Mean Square Error (RMSE) = 0.595. In Software enhancement effort estimation using correlation-based feature selection and stacking ensemble method Sakhrawi et al. [17] incorporated two models. The first model uses GBReg (Gradient Boosting Regressor), Linear SVR, M5P and RFR. The second model uses all algorithms in the first model with the exception of M5P. They used the same evaluation metrics as used in their previous study [16] but this time results show that the software estimation using Correlation-based feature selection is improved and the second has better accuracy than the first one with M5P. Chukhray et al. [18] selected and trained weak predictors in the first stage- support vector machine, K-nearest neighbour classifier and multivalued linear regression models. Random forest was employed in the boosting ensemble, which uses the boosting algorithm to integrate these inconsistent results across several iterations into a single strong prediction result. They used R-squared error, MAE, and RMSE as evaluation metrics. The new stacking model, which is built on machine learning techniques and uses random forest as a meta-algorithm, is demonstrated through this experimental study to have a minimum of 1.03 times greater RMSE than competing models. Suresh et al. [19] proposed a pragmatic ensemble learning approach for effective software effort estimation where the performance is compared to regression models such as K-nearest neighbour, stochastic gradient descent, decision tree, random forest regressor, bagging regressor, gradient boosting and Ada-boost regressor. MAE, MSE, RMSE, and R^2 were used as evaluation metrics. Their findings show that the gradient boosting regressor model is effective, as proven by its accuracy of 98% with the COCOMO'81 dataset and 93% with the China dataset. The research by Alzahami and Khan [20] uses BestFit and Genetic Algorithm for feature selection and bagging with base learners- SMOReg, Linear regression, MLP, REPTree, random forest, and M5Rule. They use MMRE and PRED (25), (50) and (75) as evaluation metrics. According to the results, the bagging M5 rule feature selection using a genetic algorithm is the best

approach for forecasting attempts with an MMRE value of 10%. PRED (25), (50) and (75) correspondingly have values of 97%, 98%, and 99%. Pospieszny et al. [21] utilised Neural Networks, Support Vector Machines, and Generalised Linear Models for their software project effort and duration estimation research. They used various evaluation metrics such as MAE, MSE, RMSE, R^2, MMRE, PRED among others and came to the conclusion that the results of the ensemble models are promising and accurate in comparison to other models and these ensemble models are appropriate for deployment. Research by Rijwani and Jain [22] use Artificial Neural Networks, Back Propagation training and Multi Layered Feed Forward Neural Network. They use MSE and MMRE as evaluation metrics. It was noted that the effort estimates produced by the neural network model were noticeably better than those produced by the widely used algorithmic model COCOMOII. The capacity to combine traditional mathematical models and expert knowledge in a common architecture, which has broad applicability in software cost estimation, was another excellent advancement of employing neural network models. Sree and Rao [23] use Adaptive Neuro Fuzzy Inference System (ANFIS), Fuzzy Logic, Neural Networks (NN), Support Vector Machines (SVM) and Random Forest in their research. The ANFIS Model provided better results compared to other models. Hidmi and Sakar [24] apply the algorithm on desharnais and maxwell datasets and use K-nearest neighbours and support vector machine getting an accuracy of 91.35% on desharnais and 85.48% on maxwell dataset. Kaushik et al. [25] applied deep belief network (DBN) with antlion optimization (ALO) technique for effort estimation in both agile as well as non-agile software development environments. Their approach is validated on four non-agile datasets and three agile datasets. They concluded that DBN-ALO provided good results for both agile and non-agile development projects. Also, Kaushik et al. [26] used whale optimization algorithm (WOA) for fine tuning the parameters of deep belief networks (DBN). They applied their technique on four software effort estimation datasets. Their technique provided better results in comparison to the technique, where DBN is fine-tuned with back propagation.

Jose Thiago H. de A. Cabral and Adriano L.I. Oliveira [27], propose dynamic ensemble selection (DES) models for software effort estimation which is heterogeneous and composed of regressors and classifiers. In the study, the regression models are trained using training data. The appropriate regression model is chosen by the trained classifier. The final prediction is given by merging chosen regressors chosen by the classifiers. The results demonstrated that the combination of regressors outperforms the individual regressors.

Jose Thiago H. de Cabral et al. [28] presented a literature review for ensemble effort estimation from 2016 to 2020. They concluded that machine learning is widely used in constructing ensemble effort estimation models. Also, the ensemble models outperform the individual models. They explored the research opportunities and found that the ensemble dynamic selection models have wider scope.

Abnane et al. [29] propose ensemble imputation technique for missing data in software effort estimation dataset. This technique highly improved estimation accuracy. The study proposes 11 heterogeneous imputation techniques which are evaluated over 6 datasets. The authors use K-nearest neighbors, expectation maximization, support vector regression (SVR) and decision trees (DTs) techniques to construct ensembled imputation techniques. The authors concluded that the ensembled imputation demonstrated at par performance than the single imputation technique.

Syed Sarmad Ali et al. [30], frame an ensembled effort estimation model by combining Use Case Point, Expert Judgment (EJ), and Artificial Neural Network (ANN). They used linear combination rule to combine these models. Their model is evaluated on International Software Benchmarking Standards Group (ISBSG) dataset. The authors concluded that ensembled technique provided better results than standalone techniques.

Wasiur Rhmann et al. [31], propose weighted ensemble of hybrid search-based algorithms using firefly algorithm, black hole optimization, and genetic algorithm. The authors compared their techniques with commonly known ML algorithms and ML based ensemble techniques. They found that weighted ensembles of hybrid search-based algorithms based on metaheuristic techniques performed at par.

Rhmann [32] provided four hybrid ensembled techniques for effort estimation. The techniques used are fuzzy and random sets-based modeling (FRSBM-R), symbolic fuzzy learning based on genetic programming (GFS-GP-R), symbolic fuzzy learning based on genetic programming grammar operators and simulated annealing (GFS_GSP_R), and least mean squares linear regression (LinearLMS_R). The author concluded that the ensembled hybrid search-based algorithm outperformed the machine learning-based ensemble bagging, vote, and stacking.

Ajay Jaiswal et al. [33], propose A Hybrid Cost Estimation Method for Planning Software Projects Using Fuzzy Logic and Machine Learning. The study uses specific datasets (Desharnais, Kitchenham, and Maxwell), which might not represent the diverse range of software projects encountered in practice. This could limit the generalizability of the results. Additionaly, Combining fuzzy logic with machine learning models can be computationally expensive. The Ensemble model achieved a maximum R-squared error of 0.9307893 in the Kitchenham dataset and had a Root Mean Squared Error of 0.2707119. Akshay Jadhav et al. [34], propose Multi-Step Dynamic Ensemble Selection to Estimate Software Effort. In their first approach using KNORA, as the number of features increases, the distance metrics used by K-nearest neighbor

algorithms become less effective. This phenomenon, known as the curse of dimensionality, can lead to poor performance in high-dimensional spaces. In their second approach using DES methods, there were two limitations. One, sensitivity to parameter tuning wherein, a careful tuning of several parameters is required which makes it more susceptible to improper tuning and thus, can lead to suboptimal performance. Two, DES methods assume that the local competence of models around the new instance is indicative of their global competence. This locality assumption can fail in cases where the data distribution is non-uniform or when there are abrupt changes in the data characteristics.

Anca-Elena Iordan [35] proposed an optimized LSTM neural network for estimation of software development effort. The study has certain drawbacks, particularly for small datasets with low dimensionality and few observations, as Albrecht and Kemerer. Future research is required to determine the cause of this mismatch and the best optimization strategy for searching tiny datasets. Wasiur Rhmann et al. [36] proposed hybrid search-based ensemble technique based on metaheuristic algorithm. The incorporation of several project types in the datasets does not eliminate the low but persistent threat to external validity. This implies that the findings might not apply to all software projects, particularly those that fall beyond the purview of the less number of datasets that were used.

The above-mentioned studies posit the motivation behind this paper, but the extent of possible studies is not just limited to these techniques. There is immense scope of new techniques that can be developed in the future with higher accuracies and better results. However, these improvements are based on the current work being carried out which is, in and of itself, an improvement with respect to the previous works.

3 Background techniques

In this section all the techniques used in the SENSE model are detailed. Stacking in subSect. 3.1, Ridge Regression, Random Forest, LightGBM and K-nearest neighbours are explained in subSect. 3.2, 3.3, 3.4 and 3.5 respectively.

3.1 Stacking

Stacking, sometimes called stacked generalisation, was first posited in 1992 by Wolpert [37]. It is a method of passing data from one group of generalisers to another before making the final prediction. Stacking is a technique wherein; various models of machine learning are ensembled together. In stacking, the various models of machine learning are not learning the entire lexicon of the problem but rather only a certain extent of it. These different models of machine learning are applied to the same problem together. Each model of machine learning applied generates an intermediate prediction, and upon adding another model, the latter learns from the intermediate predictions of the former. The last model of machine learning that learns from the intermediate predictions of the previous models can be visualised as being stacked atop the previous intermediate models.

3.2 Ridge regression

Linear regression [38] is a popular statistical machine-learning model based on supervised learning. It is used for the predictive analysis of quantifiable values. It is called linear regression since it depicts a linear correlation between a dependent variable and more than one independent variable. It shows how, in accordance with independent variables, the dependent variable changes. In this study, the use of ridge regression has been made for the purpose of tuning the model in order to examine data that suffer from multicollinearity. Due to multicollinearity, we have more variance and unbiasedness, which leads the predictions to be farther away from the values that are actual. Therefore, through ridge regression [39], a minuscule amount of bias is introduced, resulting in a notable reduction in variance, portending to improved predictions.

3.3 Random forest

One of the types of supervised machine learning algorithms are tree-based algorithms. In these algorithms, the predictions are usually made on a dataset that is in tabular form. Random Forest is an example of a tree-based algorithm. In Random Forest Regressor [40], the model of machine learning deploys the same algorithm on different subsets of the data. Random Forest is the name given to the technique where the bagging method is implemented on Decision trees. This technique lays the issue of overfitting to rest by devising a 'forest', gathering several decision trees together. The output of every decision tree is considered, and the final prediction results either from the predominant output of the decision trees or an average of the outputs.

3.4 LightGBM

By training a series of weak models that successively improve on the shortcomings of the one before it, boosting algorithm [41] aims to increase the predictive capability. Several base machine learning algorithms are utilised to create predictive models which are weak. The weak models are merged, after many successive repetitions by the boosting algorithm, into a predictive model which is strong. Gradient Boosting is one of the types of boosting algorithms. The primary concept underlying this method is to create models in succession

while attempting to minimise the flaws of the prior model accomplished by establishing a new model on the mistakes of the prior one. Gradient boosting technique [42] involving the use of decision trees for weak learners is termed as LightGBM. By using an autonomous feature selection and concentrating on boosting instances with greater gradients, LightGBM expands the gradient boosting technique. LightGBM is applied based on Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) [43].

Gradient-based One-Side Sampling considers the training samples whose gradients are large to heighten the accuracy and decrease the complexity. Exclusive Feature Bundling bundles the mutually exclusive feature with a sparse value changing the complexity of histogram thereby the speed is increased without compromising the accuracy. Together, they enable the model to function well and provide it an advantage over competing gradient boosting algorithms.

3.5 K- nearest neighbour

In this study, the K-nearest neighbour algorithm [44] is used as a classification algorithm. K-nearest neighbours (KNN) is a supervised learning technique that may be utilised for classification and regression. KNN tries to forecast the appropriate class for the test data by calculating the distance between all the training points and the test data. The KNN classifies newly input data based on its similarity to previously trained data and organises the data into clusters or subsets.

4 Dataset description and preprocessing

Section 4 describes the datasets used in this study in subSect. 4.1 and the data preprocessing methods in subSect. 4.2.

4.1 Dataset description

The use of six datasets has been made for the purpose of evaluating the SENSE model. The datasets are China, Maxwell, ISBSG, Nasa93, Albrecht and Kemerer as shown in Table 1.

The China dataset [45] contains records of 499 projects and has 18 attributes. In this paper 9 out of 18 attributes have been selected. The Albrecht dataset [46] has the records of 24 projects in an 8-attribute table.

Out of these 8 attributes 6 have been selected. The Maxwell dataset [47] incorporates 62 projects' records and has a total of 27 attributes and 9 have been selected among them. The Kemerer dataset [48] has records from 15 projects and 7 attributes. 5 attributes were selected among them. The ISBSG dataset [49] contains more than 1000 samples and 5 out of 130 attributes have been selected for further processing. The Nasa93 dataset has records from various across 93 projects of NASA. It has 17 attributes, out of which 5 have been selected.

4.2 Preprocessing

Machine learning povides learning and developing an understanding on a machine or computer level by various algorithms. This happens when the algorithms extract certain attributes or characteristics of the data from which 'learning' has to be done. For the computer to be able to understand these characteristics, they must be within the lexicon of computer understanding, which is numbers. Therefore, data preprocessing in machine learning is the conversion of unprocessed characteristics of data into a form which enables learning and understanding by the algorithm. Data preprocessing involves multiple tasks such as exploratory data analysis, feature engineering and hyperparameter tuning. Preprocessing in this study is detailed in below subsections.

4.2.1 Exploratory data analysis

Exploratory data analysis is the procedure used to interpret the available raw data. Understanding the intuition behind the data, what it represents, checking for the presence of aberrations and studying the basic premise of the data and thereafter, putting forward graphical and statistical analysis.

For this study, first the data is checked whether it contains any null and/or missing values using the Pandas library [50] in python [51]. It is observed that out of all the datasets used in the study the ISBSG dataset contained a few null/missing values in some features. These were then categorized on the basis of their feature definition, feature correlation/importance and type of data in these columns. Resultantly, the null/missing values in least important columns were simply dropped. And the null/missing values in relevant columns were imputed by computing the average(median) of non- null values.

Second, the df.describe() function is used which provides a statistical insight into the data showing the count, minima, maxima, mean, standard deviation and the 25th, 50th and 75th percentile points in the data. This statistical information helps in understanding the range of outliers for the particular dataset. Third, for further visualisation, seaborn library [52, 53] provides a means for feature correlation in the form of a heatmap. Correlation is a tool which further informs about which features can be dropped. Darker shades represent a stronger or positive correlation and lighter shades represent weaker or negative correlations. The stronger shades are usually kept, and the weaker ones are dropped.

Table 1 Description of dataset records

Data set name	Features	Count of projects	Effort data				
			Min	Max	Mean	Median	Std deviation
China	18	499	26	54620	3921.05	1829	6480.36
Albreeht	8	24	0.5	105.2	21.875	11.45	28.42
Maxwell	27	62	583	63694	3223.21	5139.5	10499.9
Ke merer	7	15	23.2	1107.31	219.25	130.3	263.06
Nasa93	17	93	8.4	8211	624.41	252	1135.93
ISBSG	130	6760	8	150040	4963.67	193S	10413.32

4.2.2 Feature engineering

After exploratory data analysis, feature engineering is carried out. Any tangible input that may be employed in a predictive model is referred to as a 'feature'. The power of an engine, number of lines of code, duration required to complete a task et cetera, are all instances of a 'feature'. The procedure of choosing, altering, and converting original data into features that may be utilised in the model's learning is known as feature engineering. To enable machine learning to operate effectively on tasks that are new, better features may need to be developed and trained. It has the potential to generate new features with the objective of streamlining and accelerating data transformations while simultaneously improving accuracy of the model. Transformation, Scaling and Feature Selection are the procedures involved in feature engineering.

- **Transformation**—The skewness of the data is eliminated via log transformation [54]. Data that deviates from the symmetric bell—shaped curve are said to be skewed. Right-skewed data are frequently transformed using log transformation. We apply transformation to the portions of our dataset with skew values greater than 1.0.
- **Scaling**—By subtracting the median and afterwards dividing by the inter—quartile (75% value–25% value), a tool in the Sci-Kit Learn library [55], Robust Scaler [56] modifies the feature vector. Unlike MinMaxScaler, the data is not scaled into a specified interval by Robust Scaler. The exact definition of the scale is not met. After Robust Scaler is used, the range for each feature is greater than it was with MinMaxScaler. Compared to MinMaxScaler, Robust Scaler was employed to decrease the impact of outliers.
- **Feature selection—**Features in a project of machine learning are the variables that have been input in the project. The dataset's columns each represent a feature. The process of picking just the characteristics that have a substantial impact on the output of the machine learning model and discarding the features that have little or no impact is known as feature selection. This is a process which involves Feature Correlation, Feature Importance and Feature Selection [57]. Feature Correlation heatmap in Fig. 3 has been used to draw out the relation between a pair of characteristics of data items within the dataset. Feature Importance using Random Forest algorithm, as shown in Eq. (1), considers the relevancy of features. Random forest has a couple hundred decision trees that have a few observations of data items achieved randomly. In Eq. (1), the value of feature importance in a random forest is calculated by averaging it over T, the total trees' number. In the equation, $RFfj_j$ is the significance of feature j deliberated from the trees in random forest model and $normfj_{jk}$,in tree k is normalized feature importance for j.

$$RFfj_j = \frac{\sum_{k \in alltrees} normfj_{jk}}{T} \tag{1}$$

4.2.3 Hyperparameter tuning

After concluding the feature engineering process, hyperparameter tuning is done. The information that controls the training process itself is included in hyperparameters. These hyperparameters have no direct relationship with the training data. Determining the hyperparameters of an algorithm in order to procure a level of performance which works in the most efficient way on a validation set is the objective of hyperparameter tuning.

In order to select the most promising hyperparameters to evaluate the objective function, which is the validation error of a model utilizing hyperparameters, Bayesian hyperparameter tuning [58] is an effective tool for optimization of costly or expensive functions. Either the maximized value is to be found for a costly function or the minimized objective function score, such as for RMSE. As opposed to grid or random search, Bayesian techniques retain note of previous assessment outcomes, which they use to build a probabilistic model linking hyperparameters to a probability of a score on the objective function [59]. Bayesian hyperparameter optimization is based on Bayes' rule which helps in predicting an event Y, given an event X denoted as $P(Y|X)$. Application of

Bayes' rule for optimization of hyperparameters is depicted in Eq. (2),

$$P(V|H) = \frac{P(H|V)P(V)}{P(H)} \tag{2}$$

where *P(V|H)* is the probability of the value to be maximized/minimized given the set of hyperparameters, *P(H|V)* is the probability of a particular set of hyperparameter when the given value is maximized/minimized. *P(V)* is the initial value and *P(H)* is the probability of receiving that particular set of hyperparameters.

5 Proposed work

In this section, the proposed SENSE model is explained. Figure 1 depicts the block diagram of the steps involved in this study, along with a detailed overview in Table 2, and further explanation mentioned below. As mentioned, in Table 2, we have configured the stacking model upon K-fold cross validation, using SVR, LightGBM, Random Forest and K- nearest neighbours as base (level-0) models and Ridge Regression as meta (level-1) models. The reason for this selection is to create each layer having a specific purpose and adding to the overall functionality and strength of the structure. Base models in this study are particularly diverse, ranging from classification to regression. Each base model algorithm has a distinct and nuanced way of interpreting the patterns in the data and thus, forwarding these results as inputs to the next level. Meta model acts as the aggregator which combines the inputs received from the diverse base models and learning the weaknesses and strengths of predictions of the base models leading to better final outputs.

For the purpose of explaining the methodology of this study in a detailed and quantitative manner, the China dataset [45] has been taken as a suitable example. When the China dataset was loaded into the pandas [50] dataframe, it consisted of 18 features (columns).

Out of the 18 columns, 'ID' and 'Dev.type' were redundant and removed at the beginning of exploratory data analysis. Before removing the skewness of features with skew values greater than two, there were some significant features with exceptionally high skewness like– 'AFP' with a skew value of 9.80, 'Input' with skew value 14.38 and 'File' with 7.47 skew value.

Upon the removal of skewness using log transformation [54] there was a drastic reduction in the skew values, so much so that all the features in the dataset had a skewness between − 1 and 1. Ideally, the range of skewness should lie between − 0.5 and 0.5. In this study, because of the use of log transformation, a substantial majority of features in the China dataset had skew values in the ideal range thus, effectively removing outliers in the data.

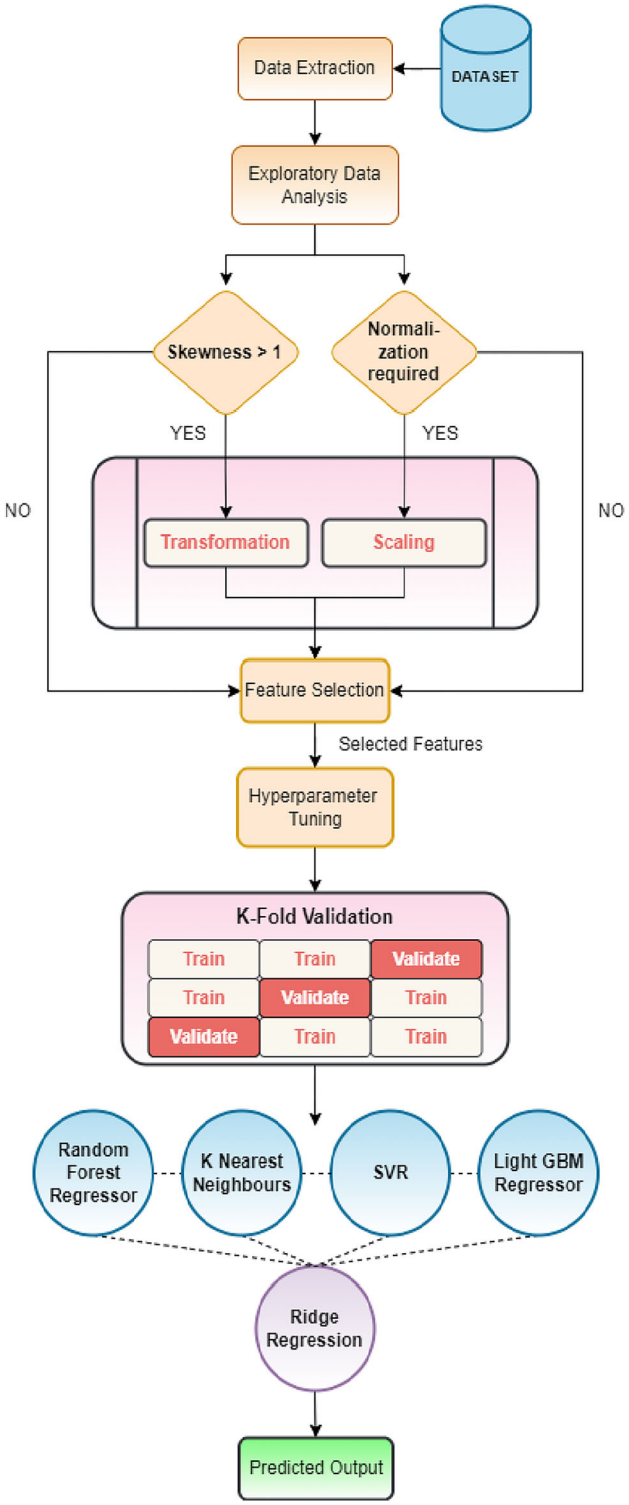

Fig. 1 Representation of methodology

To normalise the features, scaling was performed using the RobustScaler() function [56] bringing all values of the

Table 2 Proposed SENSE framework

Step: 1	Data is extracted from a file with.arff format and loaded into the dataframe of pandas library
Step: 2	To interpret and understand raw data Exploratory data analysis is performed in the following steps:
Step: 2.1	Checking and handling null values in the entire dataframe
Step: 2.2	Eliminating the skewness with value > 1, if the need exists, via log transformation
Step: 2.3	Establishing whether a need to normalise the data exists, performing scaling to normalise the data and deciding which scaling algorithm should be used
Step: 2.4	Using df.describe() to delineate the following statistics regarding the dataset- count, minima, maxima, mean, standard deviation and the 25th, 50th and 75th percentile points in the data
Step: 3	Depicting correlation between various pairs of characteristics of the dataset using heatmap as means to visualise the data
Step: 4	Considering the relevance of the characteristics of the dataset using random forest for feature importance
Step: 5	Selecting characteristics or columns of the dataset which have a substantial impact on the output and discarding redundant or insignificant columns for the purpose of feature selection
Step: 6	Splitting the dataset in training data and testing data in 80:20 ratio respectively
Step: 7	Hyperparameter tuning of the base and meta models of the stacking algorithm using BayesSearchCV() function [46] as shown in Table 3
Step: 8	Configuring the stacking model upon K-fold cross validation, using SVR, LightGBM, Random Forest and K- nearest neighbours as base (level-0) models and Ridge Regression as meta (level-1) models
Step: 9	The proposed model is evaluated on MMRE, R^2, MAE, PRED and RMSE evaluation metrics

features in the desired range. The data is scaled between its 25th and 75th percentile, i.e., the interquartile range, removing the median.

Statistical description of the dataset was obtained using the df.describe() function. Random forest enabled feature importance in Fig. 2 is used to understand the relevance of the dataset's features. A feature correlation heatmap in Fig. 3 depicts the correlation between different features of the dataset. Feature correlation and feature importance [57] selected the final, most impactful features in the dataset. These features were– 'AFP', 'Input', 'File', 'Added', 'PDR_AFP', 'PDR_UFP', 'NPDR_AFP' and 'NPDU_UFP'.

After feature selection, hyperparameter tuning was performed in order to determine those parameters of an algorithm which have the highest level of performance. The hyperparameter tuning is performed using BayesSearchCV() function [60]. This function has parameters- 'estimator',

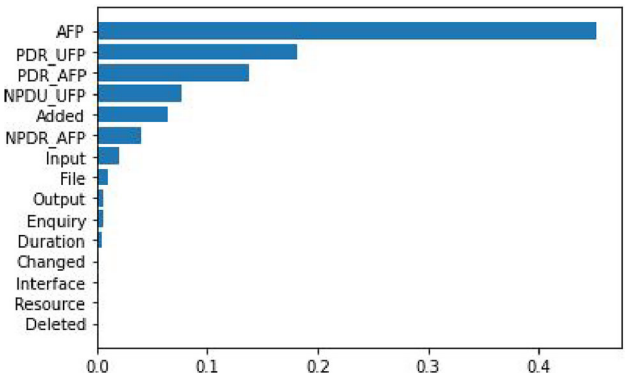

Fig. 2 Feature importance

and 'search_spaces'. The 'estimator' refers to an object or instance of a machine learning algorithm. The 'search space' specifies the range or possible values that each hyperparameter can take during the optimization process.

Hyperparameter tuning using BayesSearchCV() function [60] also gives the value of the selected parameter from the predefined range, as depicted in Table 3. In the base layer of the stacking model in this study, SVR, Random Forest regressor, K-neighbours neighbours and Light GBM algorithms are used, and Ridge Regression in meta layer.

Parameters for Random Forest Regressor [61] were, *max_features* set at a value 8– which, when finding the most appropriate split, gives information about the number of features; *min_samples_leaf* set at the default value 1– giving the least number of samples needed in order to be present at the leaf node and has a smoothening effect on the model; *min_samples_split* with the default value 2– which gives the very minimum of samples needed to separate the internal nodes; *criterion*– it is the function to assess a split's quality, for the mean squared error 'squared_error' is the criteria that is supported which reduces the L2 loss, and *n_estimators* set at the default value 100, is nothing but the least number of trees in the forest.

The *max_depth* parameter is set at 16. It is the paramount depth of the tree for base learners, which, when set at less than or equal to 0 has no limit; *learning_rate* set at 0.05071, is the rate of learning for boosting algorithm; *n_estimators* with the value 309 defines the quantity of boosted trees to be fitted, and *num_leaves* set at 2090, specify the greatest number of tree leaves for the base learners. All these parameters were used as a part of the Light GBM algorithm [62].

In the SVR algorithm [62], the parameters determined were, *C* with a value of 41.19– a regularisation parameter, which has a positive value only and regularisation's strength happens to be inversely proportional to *C*; *epsilon* with a 0 value– which establishes a tolerance margin without any penalty in relation to the training loss function and it is also definitely positive, and *kernel* – which defines the type of

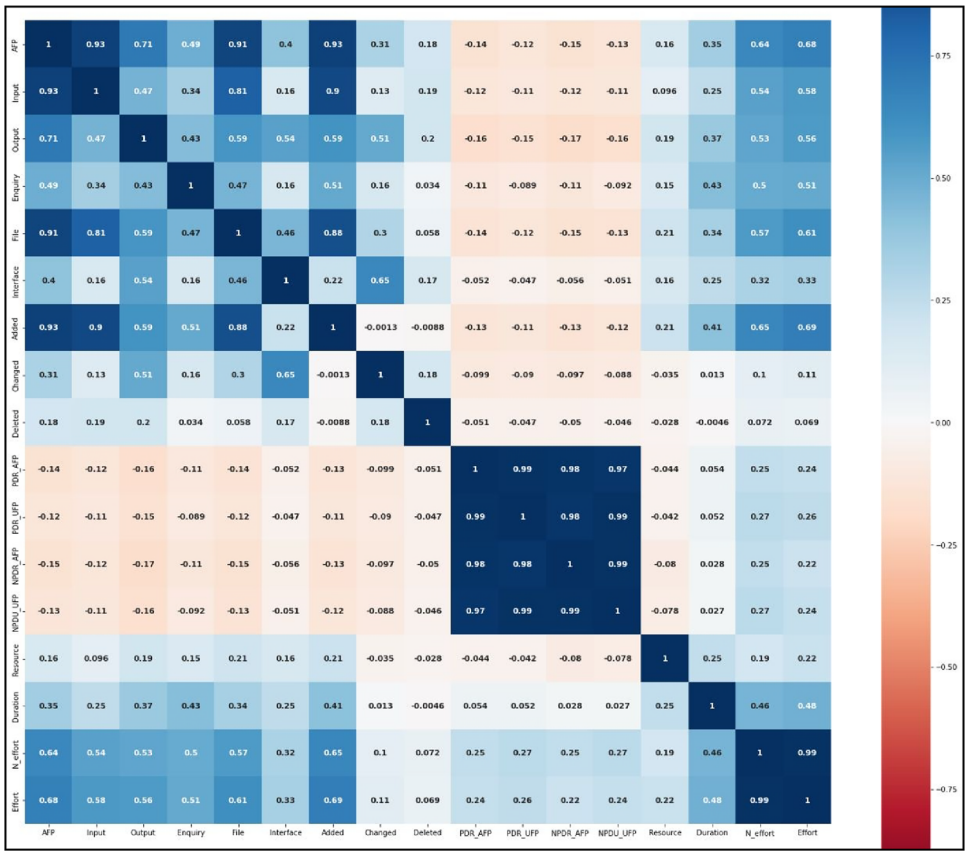

Fig. 3 Feature correlation Heatmap

kernel deemed appropriate for use in the algorithm, in this study the default 'rbf' kernel was used.

The *n_neighbours* parameter set at the value of 5 was used in the K-nearest neighbours [64] algorithm. For K-neighbours queries, this parameter describes the number of neighbours to be used by default.

In the meta layer, for the Ridge Regression algorithm [65], parameters determined were- *alpha* with value of 0.00494– which, while controlling the strength of regularisation, must be a positive float; *fit_intercept* with boolean value True– dictating that the intercept is to be fitted for the model in this study, and *solver* set as 'lsqr', which makes use of iterative procedures and it is the quickest to use in routines for computation.

When hyperparameter tuning is complete, these parameters are put in the stacking regressor function. K-fold cross validation [66] is performed in the stacking regressor. K-fold cross validation is when the dataset is split into K equal sized groups. These groups are used to train and test the model K number of times. The portions used to train and test the models are changed once during each iteration until you reach the Kth-iteration. In stacking regressor K = 3 is used. Each time, the portions of the dataset are changed while training and testing during each one of the K iterations. After the stacking model is run, the output is evaluated on MMRE, R2, MAE, PRED and RMSE evaluation metrics detailed below in Sect. 6.

6 Evaluation criteria

The following extensively used evaluation criteria have been employed to analyse and compare the accuracy of the SENSE model.

The difference between a quantity's estimated value and its actual value is known as the absolute error (AE). Absolute error, in Eq. (3), is insufficient when used as a standalone quantity because it provides no information about the significance of the error. But it is still extensively used in the calculation of various other evaluation metrics.

$$AE_j = \left|\gamma_j - \widehat{\gamma}_j\right| \tag{3}$$

In the above equation, AE_j is the absolute error, γ_j is the estimated value and $\widehat{\gamma}_j$ is the actual or original value.

Table 3 Obtained Hyperparameter values

Models used		China	Albreeht	Maxwell	Kemerer	Nasa93	ISBSG
Random forest	criterion	squared_error	squared_error	squared_error	squared_error	squared_error	absolute_error
	max features	8	1	5	2	4	4
	min_samples_leaf	1	1	1	1	2	1
	min_samples_split	2	2	2	3	2	2
	n estimators	100	200	163	200	1640	2000
Light GBM	learning_rate	0.05 071	0.0S405	0.01947	0.61079	1.30099	0.03179
	maxdepth	I6	39	36	42	46	25
	min_data_in_leaf	13	3	4	3	18	0
	nestimators	309	37	450	31	461	2000
	num leaves	2090	2031	2363	1099	1445	20
SVR	C	41.19937	3.23919	39.24395	43.69557	100.00000	S.4035S
	epsilon	0	0	0.94164	0.0S373	0.60867	0.42375
	kernel	rbf	rbf	poly	poly	rbf	rbf
KNN	nneighbors	3	2	2	2	6	3
Ridge	alpha	0.00494	1.533S6	0.00494	0.04597	0.04597	0.00494
	fitintercept	True	True	True	True	True	True
	normalize	True	True	True	True	True	True
	solver	Isqr	saga	Isqr	Isqr	Isqr	Isqr

The average of absolute errors of every test sample is given the name of Mean Absolute Error (MAE) [67]. In Eq. (4), k is the number of test samples and j is the iteration number.

$$MAE = \frac{1}{k} \times \sum_{j=1}^{k} AE_j \tag{4}$$

Root Mean Square Error (RMSE) is measured by considering the residuals (distance between actual or real value and estimated values) and the standard deviation (how the values are disseminated across the model). Therefore, the residuals replace the variance in the formula of standard deviation in order to calculate RMSE shown in Eq. (5). However, using the MAE is more advantageous than using RMSE [68].

$$RMSE = \frac{\sqrt{\sum_{j=1}^{k} (\gamma_j - \widehat{\gamma})}}{k} \tag{5}$$

R-squared [69] measure is a quantity which informs about the appropriateness of the model's fitting by comparing the sum of squares of the differences between estimated value and original or actual value with the sum of squares of the differences between the estimated value and the average of actual values. The former is called Residual Sum of Squares (RSS) and the latter is called Total Sum of Squares (TSS) as depicted in Eq. (6). R-squared error has the most ideal value of 1. The model is better fitted if the R-squared value is closer to 1.

$$R^2 = 1 - \frac{RSS}{TSS} \tag{6}$$

MRE is kenned as the magnitude of relative error. It measures the percentage of total error to the original or actual value of effort. It is calculated by the dividing AE with the actual or original value of effort γ_j in Eq. (7).

$$MRE_j = \frac{AE_j}{\gamma_j} \tag{7}$$

PRED provides the prediction of the output. Quantised by the prediction percentage that lies within the original known value of the effort L. If, MRE is less than or equal to 0.25 then its PRED is always 100%. The formula for discerning L_j is given in Eq. (8) and k is number of test samples in Eq. (9).

$$L_j = \begin{cases} 1 & \text{if } MRE_j \leq 0.25 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$PRED(g) = \frac{100}{k} \sum_{j=1}^{k} L_j \tag{9}$$

MMRE is the average of MRE for k test samples as shown in Eq. (10).

$$MMRE = \frac{1}{k} \sum_{j=1}^{k} MRE_j \tag{10}$$

Table 4 Results of the proposed approach SENSE

DATASET	MAE	RMSE	MMRE	R2	PRED (%)
China	0.043774	0.0125316	0.00516793	0.9391	100
Albreeht	0.0299979	0.0329386	0.135391003	0.9037	95.33
Maxwell	0.0405917	0.0487294	0.03105203	0.3199	100
KemerEr	0.0215404	0.0243463	0.051326434	0.3366	100
Nasa93	0.0466537	0.0642637	0.053351945	0.7055	96.77
ISBSG	1.010623	0.0137205	0.006003036	0.9399	100

7 Experimental evaluation

In this section experimental results of the study are discussed. Table 4 lists the results of the technique proposed by the SENSE model of this study on the datasets China, Albrecht, Maxwell, Kemerer, Nasa93 and ISBSG. These results are compared with the earlier results of ensembling methods by the authors [14, 15, 17, 19, 21]. All the earlier studies referenced are not using all the datasets. Few authors have evaluated the results on China and Cocomo81 datasets and few others only on NASA93 and so on. Table 5 depicts results of the China dataset on three evaluation criteria MAE, RMSE and R^2. It contains the results of ensemble approaches, averaging, weighted averaging, bagging, boosting, and stacking using RF presented in the work of Priya Varshini et al. [14]. The results are also equated with regression models such as bagging regressor (BR), decision tree (DT), stochastic gradient descent (SGD), K-nearest neighbour (KNN), random forest regressor (RFR), gradient boosting regressor (GBR), and Ada-boost regression (ABR) given by Suresh Kumar et al. [19]. The results of the authors [19] are normalised between (0–1) in order to have a clearer understanding and comparison of the techniques. Ahmad and Ibrahim [70] proposed Long Short Term Memory (LSTM) model for software development effort estimation. Their results are also listed along with other results.

Tables 6, 7, 8 are demonstrating results on Albrecht, Kemerer and Maxwell datasets. These tables contain results contributed by Priya Varshini et al. [14] and SENSE model. When these results are compared, the results of the SENSE model are found to be more promising.

Table 9 shows results of the ISBSG dataset on five evaluation criteria– MAE, RMSE, R^2, MMRE and PRED. All the studies [15, 17, 21] in this table are not evaluated on all these five metrics but, rather, different studies use different combinations of the evaluation metrics as listed in Table 9.

The study conducted by Kumar and Venkatesan [15] presented a model based on linear regression, random forest, neural networks and stacking algorithm, with results evaluated on MAE, MMRE and PRED. Sakhvari et al. [17] evaluated the results of M5P [71] and stacking algorithms on MAE, RMSE and R^2. Support Vector Machine (SVM), Multilayer Perceptron (MLP), Generalised Linear Model (GLM) and ensemble algorithms have been evaluated on MAE, RMSE, MMRE and PRED by Pospieszny et al. [21].

Table 5 Results on China dataset

Approach	Evaluation criteria		
	MAE	RMSE	R2
Averaging [14]	0.0199	0.0611	0.S263
Weighted Averaging [14]	0.0530	0.1214	0.3153
Bagging [14]	0.0120	0.0412	0.9211
Boosting [14]	0.0105	0.0361	0.9478
Stacked Random Forest [14]	0.0040	0.0156	0.9839
SGD[19]	0.0385	0.0828	0.6000
KNN [19]	0.0415	0.0816	0.6100
DT[19]	0.0232	0.0460	0.8700
BR [19]	0.0165	0.0440	0.8800
RFR[19]	0.0148	0.0419	0.8900
ABR[19]	0.0271	0.0499	0.S500
GBR [19]	0.0124	0.0330	0.9300
LSTM [70]	0.0160	0.0190	0.9720
SENSE (Proposed Approach)	0.0437	0.0125	0.9891

Table 6 Results on Albrecht dataset

Approach	Evaluation criteria		
	MAE	RMSE	R2
Averaging [14]	0.2133	0.2403	0.4109
Weighted Averaging [14]	0.165S	0.2789	0.2066
Bagging [14]	0.1784	0.2264	0.4773
Boasting [14]	0.1237	0.1843	0.6855
Stacked random forest [14]	0.0288	0.0370	0.9357
SENSE [Proposed Approach)	0.02939	0.03293	0.9087

In Figs. 4 and 5, ISBSG and China dataset results are shown, respectively, with next-to-no deviation in predicted values as compared to the actual values. Further comparing the actual and predicted values for Kemerer, Nasa93, Maxwell and Albrecht datasets, there was diminutive deviation, if any, from the actual values.

Thus, the experimental results which have been calculated on six datasets- Maxwell, China, Kemerer, Nasa93, Albrecht

Table 7 Results on Kemerer dataset

Approach	Evaluation criteria		
	MAE	RMSE	R2
Averaging [14]	0.2397	0.2719	0.4720
Weighted Averaging [14]	0.1811	0.3253	0.2444
Bagging [14]	0.2042	0.2399	0.5890
Boosting [14]	0.2345	0.2626	0.5907
Stacked random forest [14]	0.0703	0.1094	0.7520
SENSE (Proposed Approach)	0.02154	0.02434	0.S366

Table 8 Results on maxwell dataset

Approach	Evaluation criteria		
	MAE	RMSE	R2
Averaging [14]	0.0900	0.1386	0.5942
Weighted Averaging [14]	0.1211	0.2096	0.0732
Bagging [14]	0.2356	0.3120	0.0962
Boosting [14]	0.0820	0.1215	0.6880
Stacked random forest [14]	0.0356	0.0637	0.8120
SENSE (Proposed Approach)	0.04059	0.04872	0.8199

Table 9 Results on ISBSG dataset

Approach	Evaluation criteria				
	MAE	RMSE	R2	MMRE	PRED
Linear_Regression [15]	0.1500	–	–	0.1400	86% (PRED 25%)
RF [15]	0.1200	–	–	0.1100	92% (PRED 25%)
Neural network [15]	0.1800	–	–	0.1700	82% (PRED 25%)
Slacking [15]	0.1100	–	–	0.1000	92.5% (PRED 25%)
MSP [17]	0.0612	0.2514	0.9350	–	–
Stacking [17]	0.0383	0.1973	0.9370	–	–
SVM [21]	0.1900	0.2700	–	0.1300	76.91%
MLP [21]	0.2600	0.3400	–	0.2100	64.65%
GLM [21]	0.2700	0.3500	–	0.1300	61.96%
Ensemble [21]	0.2300	0.3100	–	0.1700	69.44%
SENSE (Proposed Approach)	0.0106	0.0137	0.9399	0.0060	100.00%

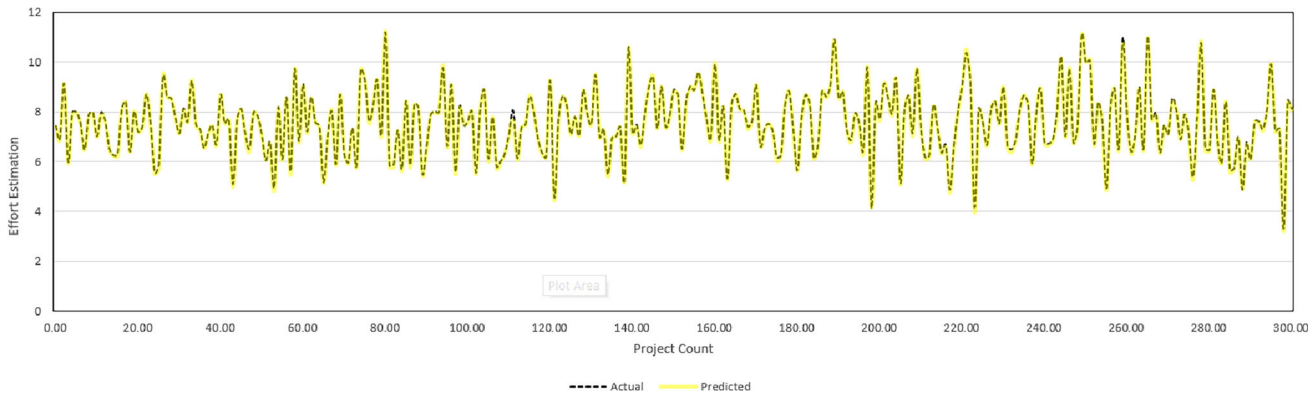

Fig. 4 Representation of ISBSG dataset actual vs predicted values

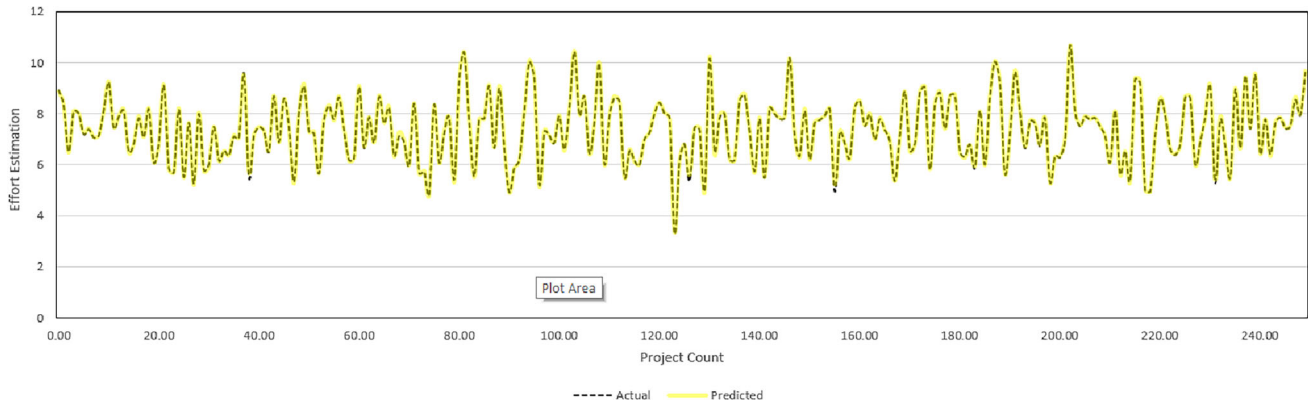

Fig. 5 Representation of China dataset actual vs predicted values

and ISBSG were subjected to MAE, RMSE and MMRE as loss indicators and R^2 and PRED as accuracy indicators, posit a promising performance. From the tables above, Table 4 shows that the ISBSG dataset produces the best result for SENSE model, for it has the highest R^2 score followed by the China dataset. Both Kemerer and ISBSG datasets have exceptionally minimal values of MAE as depicted in Tables 7 and 9. RMSE value is minimum for China dataset as demonstrated in Table 5. MMRE and RMSE values are minimum for ISBSG dataset as shown in Table 9. Maxwell, Kemerer, ISBSG and China datasets show cent percent accuracy on PRED, while Albrecht and Nasa93 have more than 95 percent accuracy on it. R^2 is considered the most significant of the evaluation metrics as it shows the extent of dependent variable's movement corresponding to the independent variable i.e. the efficiency of model fitting and is more informative than other evaluation metrics as shown in study by Chicco et al. [72]. For SENSE model, most of the datasets have an R^2 score above 90 percent, and none has an R^2 score below 70 percent reflecting the superior performance of this model.

Error analysis is a vital step towards evaluating a model's performance. The SENSE model performed well for ISBSG, China, Albrecht, and Kemerer datasets in comparison to NASA93 and Maxwell datasets. Some overfitting may have been caused due to a smaller number of samples in NASA93 and Maxwell datasets and low feature to size of sample ratio in these. The datasets ISBSG, China, Albrecht, and Kemerer often offer broader and more varied representations of software projects across regions, industries or specific metrics. They could offer a more thorough perspective for modeling and analysis, taking into account a larger variety of variables and contexts in estimating software cost. NASA93 and Maxwell datasets are comparatively older and lack, to some extent, a current reflection of software development practices.

8 Statistical analysis

Statistical analysis is an important step to conclude the research outcomes as it confirms whether we are moving in right direction or not. The statistical tests used here are non-parametric, as the effort estimation datasets do not belong to any particular distribution [73]. The tests are performed using KEEL tool [74, 75]. Initially, Friedman's test is performed to compare multiple techniques. The techniques used for comparison are averaging, weighted averaging, bagging, boosting, stacking using random forest [14] and the proposed SENSE technique. The datasets used are China, Albrecht, Kemerer and Maxwell datasets and the RMSE metric is used for conducting the test. The ISBSG dataset is not used by the authors [14] for their model evaluation.

The Friedman test equation is given by:

$$FriedmanT = \frac{(a-1)\left[\sum_{j=1}^{a}\hat{L}_j^2 - (\frac{an^2}{4})(an+1)^2\right]}{\{[an(an+1)(2an+1)]/6\} - (1/a)\sum_{i=1}^{n}\hat{L}_i^2} \tag{11}$$

where, $\widehat{L}_j$ is the total rank of jth technique, $\widehat{L}_i$ is the total rank of the ith dataset, and a is the total number of techniques. The best technique or procedure receives the lowest rank in the test. To determine the statistical variances between the techniques, the test statistic *FriedmanT* is generated. The null hypothesis made the supposition that all techniques are equivalent.

The results of Friedman test are given in Table 10. Here, the degree of freedom (df) is 5 as we have six techniques to compare, and n is the number of input datasets which is 4. The standard $\chi 2$ value with 5 df and significance value $\alpha = 0.05$ is 11.070. As the $\chi 2$ value listed in the test statistic (Table 10) is more than 11.070 and the p value is less than 0.05, our null hypothesis is rejected. So, the techniques are different.

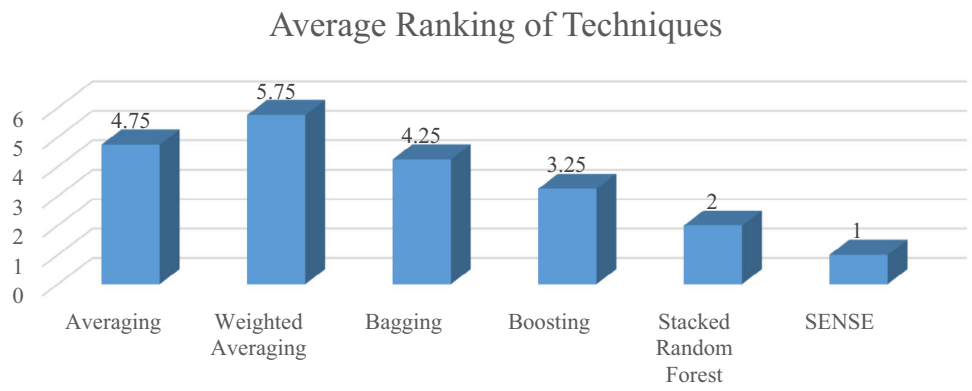

Fig. 6 Friedman rank test comparison

Table 10 Friedman rank test statistics

N	4
$\chi 2$	18
Df	5
p value	0.002946

Table 11 Holm test statistics

Techniques	Z	Holm	Hypothesis ($\alpha = 0.05$)
SENSE vs Weighted averaging	3.590662	0.01	Rejected for SENSE
SENSE vs Averaging	2.834734	0.0125	Rejected for SENSE
SENSE vs Bagging	2.456769	0.016667	Rejected for SENSE
SENSE vs Boosting	1.70084	0.025	Rejected for SENSE
SENSE vs Stacked random forest	0.755929	0.05	Not rejected for SENSE

Figure 6 depicts the average rankings of all the techniques given by the Friedman rank test. As a result of the rejection of our null hypothesis, the Holm post-hoc test [76] is used to compare the techniques, with the SENSE model having rank 1 as the control method. Table 11 shows its test statistics.

The SENSE technique performed better than all other techniques according to the Holm test statistics, except for the Stacked Random Forest technique, where the Holm value is equal to 0.05. Now, there is a need to perform a third statistical test which will compare SENSE and Stacked Random Forest technique. Wilcoxon Signed Rank Test [77] is the test used to compare the two techniques based on positive and negative rank differences. The null hypothesis presupposed an equivalence of the two techniques. The alternate theory proposed that the two methods are distinct. The test statistics are provided in Table 12.

Here, R^+ shows the sum of ranks in which the first technique SENSE performed better than the second technique Stacked Random Forest and R^- shows the sum of ranks for the opposite. The p-value is less than 0.05, so the null hypotheses is rejected, and the first algorithm SENSE outperformed Stacked Random Forest.

Table 12 Wilcoxon signed rank test statistics

Techniques	Rank positive (R^+)	Rank negative (R^-)	*p* value
SENSE vs Stacked random forest	10.0	0.0	0.04461

9 Threats to validity

The SENSE model used in the study is an ensembled approach to software effort estimation. It poses few threats to validity which are identified as below:

- The study used SVR, Random Forest, KNN, LightGBM as base learners and Ridge Regression as the meta learner. This choice of learner configuration provided us the best results for maximum datasets. The model's performance may vary if we choose some other learner configuration.
- The datasets chosen to evaluate SENSE is not biased. They were standard, open source and a conventional choice for software effort estimation studies. These are chosen for a comprehensive comparative analysis with other studies and implemented models. But the datasets play an important role in model's evaluation. The SENSE model may perform differently for the datasets other than those used in the study. Model's performance may vary if we choose some other learner configuration.
- The datasets chosen to evaluate SENSE is not biased. They were standard, open source and a conventional choice for software effort estimation studies. These are chosen for a comprehensive comparative analysis with other studies and implemented models. But the datasets play an important role in model's evaluation. The SENSE model may

perform differently for the datasets other than those used in the study.
- There is an overfitting issue with NASA 93 and MAXWELL datasets due to a smaller number of samples and low Feature to Size of Sample ratio in these. Another notable cause of overfitting is the Random Forest model which is prone to overfit with a smaller number of samples in data thus it performed well during training but not as much during testing.
- Evaluation metrics confirms the validity of a model. The evaluation metrics used in the study are commonly used by the effort estimation studies. Few other evaluation metrics can also be tried to evaluate the model.

10 Conclusion

The SENSE- Software Effort Estimation using Novel Stacking and Ensemble learning- framework proposed in this study has quite a significant differentiating factor from other stacking-based effort estimation techniques. This difference is substantiated in the selection of level-0 algorithms for the stacking model configuration. In this study, at least one algorithm is selected from the available linear, boosting ensemble, classification, and tree-based models, leading to diversification of the stacking model. SVR [63] from linear model, Random Forest regressor [61] from Tree based model, K-nearest neighbours [64] from classification model and LightGBM [62] from the boosting ensemble model have been utilized as base-models and Ridge Regression [65] which is also a linear model constitutes the meta- model layer of our proposed stacking framework. SVR and Ridge Regression have been selected from within the linear model algorithms which further evinces the diversified nature of this study.

The challenging nature of software project management and especially estimation of software effort cost will always motivate further studies in the future. The studies involving ensemble learning approaches are more accurate than others [78], especially single models as shown by Idris et al. [79]. The SENSE model is another iteration of ensemble learning methods used for software cost estimation. In the SENSE model, all the datasets—China, Maxwell, ISBSG, Kemerer, Albrecht, and Nasa93—have not only been evaluated on all five-evaluation metrics but also display more efficient results of the stacking model as compared to other studies.

Even though this study has achieved better results in software effort estimation, it would be imperative to note that scope of future work is always there to further streamline the process of estimating software cost.

A major limitation discovered in SENSE is overfitting due to Random Forest model because of which it accomplished good results in training but not in testing. This limitation can be solved using data augmentation, customizing ensemble models by using different variations of models and data regularization.

Furthermore, in addition to using conventional datasets, some real time datasets can be gathered as per the current scenario going on in software development industry to test the proposed models.

Author contributions Dr. Anupama Kaushik, gave the idea and contributed towards experimental evaluation and statistical analysis along with the manuscript review process. Dr. Kavita Sheoran, contributed towards the overall development in framing various sections and detailed review of the manuscript. Ritvik Kapur, did the python implementation and contributed towards SENSE algorithm along with the manuscript review process. Nikhil Bhutani, did the python implementation and contributed towards SENSE algorithm along with the manuscript review process. Bhavesh Singh, contributed in manuscript development framing various sections like: introduction, related work, background techniques, dataset description, part of experimental evaluation and reviewing the manuscript. Harsh Sharma, contributed towards the review process of the manuscript.

Funding All the authors declare that there is no funding used and received for the work.

Availability of data and materials The datasets used in the work are available at https://github.com/danrodgar/DASE/tree/master/datasets/effortEstimation

Declarations

Conflict of interest The authors declare no competing interests.

Ethical approval This article does not contain any studies with human participants and/or animals performed by any of the authors.

References

1. Heemstra FJ (1992) Software cost estimation. Inf Softw Technol 34:627–639
2. Boehm B, Abts C, Chulani S (2000) Software development cost estimation approaches—a survey. Ann Softw Eng 10:177–205
3. Boehm BW (1981) Software engineering economics. Prentice Hall, Englewood Cliffs
4. Putnam LH (1978) A general empirical solution to the macro software sizing and estimating problem. IEEE Trans Software Eng 4:345–361
5. Galorath DD, Evans MW (2006) Software sizing, estimation and risk management. Auerbach Publications, Boston
6. Albrecht AJ (1979) Measuring application development productivity. In: Proceedings of IBM Application Development Symposium, Monterey, California, pp. 83–92
7. Abran A, Desharnais JM, Oligny S, St-Pierre D, Symons C (2007) COSMIC- 3.0.1, Measurement Manual
8. Hodgkinson AC, Garratt PW (1999) A neuro fuzzy cost estimator. In: Proceedings of the International Conference on Software Engineering and Applications, IASTED/Acta Press, Anaheim, California, pp.401–406
9. Benala TR, Mall R (2018) DABE: differential evolution in analogy-based software development effort estimation. Swarm Evol Comput 38:158–172

10. Suresh Kumar P, Behera HS (2020) Estimating software effort using neural network: an experimental investigation. In: Das A, Nayak J, Naik B, Dutta S, Pelusi D (eds) Computational intelligence in pattern recognition advances in intelligent systems and computing. Springer, Berlin. https://doi.org/10.1007/978-981-15-2449-3_14
11. Suresh Kumar P, Behera HS, Anisha Kumari K, Nayak J, Naik B (2020) Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades. Comput Sci Rev. https://doi.org/10.1016/j.cosrev.2020.100288
12. Rama Sree P, Ramesh SNSVSC (2016) Improving efficiency of fuzzy models for effort estimation by cascading & clustering techniques. Procedia Comput Sci 85:278–285
13. Kaushik A, Kaur P, Choudhary N, Priyanka (2022) Stacking regularization in analogy-based software effort estimation. Soft Comput 26:1197–1216
14. Priya Varshini AG, Anitha Kumari K, Varadarajan V (2021) Estimating software development efforts using a random forest-based stacked ensemble approach. Electronics 10:1195
15. Sampath Kumar P, Venkatesan R (2020) Improving accuracy of software estimation using stacking ensemble method. Algorithms Intell Syst. https://doi.org/10.1007/978-981-15-5243-4_18
16. Sakhrawi Z, Sellami A, Bouassida N (2022) Software enhancement effort estimation using stacking ensemble model within the scrum projects: A proposed web interface. In: Proceedings of the 17th International Conference on Software Technologies, pp 91–100
17. Sakhrawi Z, Sellami A, Bouassida N (2021) Software enhancement effort estimation using correlation-based feature selection and stacking ensemble method. Clust Comput 25:2779–2792
18. Chukhray N, Shakhovska N, Mrykhina O, Lisovska L, Izonin I (2022) Stacking machine learning model for the assessment of R&D product's readiness and method for its cost estimation. Mathematics 10:1466
19. Suresh Kumar P, Behera HS, Nayak J, Naik B (2021) A pragmatic ensemble learning approach for effective software effort estimation. Innovations Syst Softw Eng 18:283–299
20. Alhazmi OH, Khan MZ (2020) Software effort prediction using ensemble learning methods. J Softw Eng Appl 13:143–160
21. Pospieszny P, Czarnacka-Chrobot B, Kobylinski A (2018) An effective approach for software project effort and duration estimation with machine learning algorithms. J Syst Softw 137:184–196
22. Rijwani P, Jain S (2016) Enhanced software effort estimation using multi layered feed forward artificial neural network technique. Procedia Comput Sci 89:307–312
23. Sree SR, Rao CP (2020) A study on application of soft computing techniques for software effort estimation. J Towards Bio-inspired Techn Softw Eng Springer. https://doi.org/10.1007/978-3-030-40928-9_8
24. Hidmi O, Sakar B (2017) Software development effort estimation using ensemble machine learning. Int J Comput Commun Instrument Eng 4:143–147
25. Kaushik A, Tayal DK, Yadav K (2019) A comparative analysis on effort estimation for agile and non-agile software projects using DBN-ALO. Arab J Sci Eng 45:2605–2618
26. Kaushik A, Singal N, Prasad M (2022) Incorporating whale optimization algorithm with deep belief network for software development effort estimation. Int J Syst Assurance Eng Manag 13:1637–1651
27. de Jose Thiago H, Cabral A, Oliveira ALI (2021) Ensemble Effort Estimation using dynamic selection. J Syst Softw 175:110904. https://doi.org/10.1016/j.jss.2021.110904
28. Cabral JT, Oliveira AL, da Silva FQ (2023) Ensemble effort estimation: an updated and extended systematic literature review. J Syst Softw 195:111542
29. Abnane I, Idri A, Chlioui I et al (2023) Evaluating ensemble imputation in software effort estimation. Empirical Softw Eng 28:56. https://doi.org/10.1007/s10664-022-0260-0
30. Ali SS, Ren J, Zhang K, Ji Wu, Liu C (2023) Heterogeneous ensemble model to optimize software effort estimation accuracy. IEEE Access. https://doi.org/10.1109/ACCESS.2023.3256533
31. Rhmann W, Pandey B, Ansari GA (2022) Software effort estimation using ensemble of hybrid search-based algorithms based on metaheuristic algorithms. Innovations Syst Softw Eng 18:309–319
32. Rhmann W (2021) An ensemble of hybrid search-based algorithms for software effort prediction. Int J Softw Sci Comput Intell 13(3):28–37. https://doi.org/10.4018/IJSSCI.2021070103
33. Jaiswal A, Raikwal J, Raikwal P (2023) A hybrid cost estimation method for planning software projects using fuzzy logic and machine learning. Int J Intell Syst Appl Eng 12:696–707
34. Jadhav A, Shandilya SK, Izonin I, Muzyka R (2024) Multi-step dynamic ensemble selection to estimate software effort. Appl Artif Intell. https://doi.org/10.1080/08839514.2024.2351718
35. Iordan A-E (2024) An optimized LSTM neural network for accurate estimation of software development effort. Mathematics 12(2):200. https://doi.org/10.3390/math12020200
36. Rhmann W, Pandey B, Ansari GA (2022) Software effort estimation using ensemble of hybrid search-based algorithms based on metaheuristic algorithms. Innov Syst Softw Eng 18:309–319. https://doi.org/10.1007/s11334-020-00377-0
37. Wolpert DH (1992) Stacked generalization. Neural Netw 5:241–259
38. Su X, Yan X, Tsai CL (2012) Linear regression. Wiley Interdisciplinary Rev Comput Stat 4:275–294
39. Hoerl R (2020) Ridge regression: a historical context. Technometrics 62:420–425
40. Breiman L (2001) Random forests. Mach Learn 45:5–32
41. Schapire RE (2003) The boosting approach to machine learning: an overview. In: Denison DD, Hansen MH, Holmes CC, Mallick B, Yu B (eds) Nonlinear estimation and classification lecture notes in statistics. Springer, Berlin, pp 149–171
42. Korstanje J (2021) Gradient boosting with XGBoost and lightgbm. Adv Forecasting Python. https://doi.org/10.1007/978-1-4842-7150-6_15
43. Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu TY (2017) LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In: 31st Conference on Neural Information Processing Systems, pp 3149–3157
44. Taunk K, De S, Verma S, Swetapadma A (2019) A brief review of nearest neighbor algorithm for learning and classification. In: International Conference on Intelligent Computing and Control Systems (ICCS) pp 1255–1260, https://doi.org/10.1109/ICCS45141.2019.9065747
45. Yun FH (2010) China: Effort estimation dataset. Zenodo, Switzerland, Tech.
46. Albrecht AJ, Gaffney JE (1983) Software function, source lines of code, and development effort prediction: a software science validation. IEEE Trans Softw Eng 6:639–648
47. Maxwell KD, Forselius P (2000) Benchmarking software development productivity. IEEE Softw 17:80–88
48. Kemerer CF (1987) An empirical validation of software cost estimation models. Commun ACM 30:416–429
49. Unlu H, Yalcin AG, Ozturk D, Akkaya G, Kalecik M, Ekici NU, Orhan O, Ciftci O, Yumlu S, Demirors O (2021) Software effort estimation using ISBSG Dataset: Multiple case studies. In:15th Turkish National Software Engineering Symposium pp 1–6
50. Pandas - Python Data Analysis Library. https://pandas.pydata.org/. Accessed 20 January 2023
51. Welcome to Python.org. *Python.org*, https://www.python.org/. Accessed 28 January 2023

52. Waskom M (2021) Seaborn: statistical data visualization. J Open Source Softw 6:3021
53. Seaborn: Statistical Data Visualization — Seaborn 0.12.2 Documentation, https://seaborn.pydata.org/. Accessed 5 Feb. 2023
54. Feng C, Wang H, Lu N, Chen T, He H, Lu Y, Tu XM (2014) Log-transformation and its implications for data analysis. Shanghai Arch Psychiatry 26:105–109
55. Pedregosa et al (2011) Scikit-learn: machine learning in python. JMLR 12:2825–2830
56. Sklearn.preprocessing.RobustScaler, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html Accessed 20 December 2022
57. Cai J, Luo J, Wang S, Yang S (2018) Feature selection in machine learning: a new perspective. Neurocomputing 300:70–79
58. Wu J, Chen XY, Zhang H, Xiong LD, Lei H, Deng SH (2019) Hyperparameter optimization for machine learning models based on bayesian optimization. J Electronic Sci Technol 17:26–40
59. Koehrsen W (2023) A conceptual explanation of Bayesian hyperparameter optimization for Machine Learning, https://towardsdatascience.com/a-concep tual-explanation-of-bayesian-model-bas ed-hyperparameter-optimization-for-mac hine-learning-b8172278050f. Accessed 15 January 2023.
60. Skopt.BayesSearchCV, https://scikit-optimize.github.io/stable/modules/generated/skopt.BayesSearchCV.html. Accessed 1 February 2023.
61. Sklearn.ensemble.RandomForestRegressor https://scikit-learn.org/stable/modules/ generated/sklearn.ensemble.RandomForestRegressor.html. Accessed 1 February 2023.
62. Lightgbm.LGBMRegressor,https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMRegressor.html. Accessed 1 February 2023.
63. Sklearn.svm.SVR,https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html. Accessed 1 February 2023.
64. Sklearn.neighbors.KNeighborsClassifier,https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html. Accessed 1 February 2023.
65. Sklearn.linear_model.Ridge,https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html. Accessed 1 February 2023.
66. Nti IK, Nyarko-Boateng O, Aning J (2021) Performance of machine learning algorithms with different K values in K-fold cross validation. Int J Inf Technol Comput Sci 13:61–71
67. Gergonne JD (1974) The application of the method of least squares to the interpolation of sequences. Hist Math 1:439–447
68. Willmott CJ, Matsuura K (2005) Advantages of the mean absolute error (mae) over the root mean square error (RMSE) in assessing average model performance. Climate Res 30:79–82
69. Wright S (1921) Correlation and causation. J Agric Res 20:557–585
70. Ahmad FB, Ibrahim LM (2022) Software development effort estimation techniques using Long short term memory. In: International Conference on Computer Science and Software Engineering pp. 182–187, https://doi.org/10.1109/CSASE51777.2022.9759751
71. Wang Y, Witten I H (1996) Induction of model trees for predicting continuous classes. (Working paper 96/23). Hamilton, New Zealand: University of Waikato, Department of Computer Science.
72. Chicco D, Warrens MJ, Jurman G (2021) The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. Peer J Comput Sci. https://doi.org/10.7717/peerj-cs.623
73. Kaushik A, Tayal DK, Yadav K, Kaur A (2016) Integrating firefly algorithm in artificial neural network models for accurate software cost predictions. J of Softw Evolution Process 28:665–688
74. Alcalá-Fdez J, Fernández A, Luengo J, Derrac J, García S, Sánchez L, Herrera F (2011) KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J Multiple Valued Logic Soft Comput 17:255–287
75. Kaur P, Gossain A (2019) FF-SMOTE: A metaheuristic approach to combat class imbalance in binary classification. J Appl Artif Intell 33(5):420–439
76. Holm S (1979) A simple sequentially rejective multiple test procedure. Scandinavian J Statist 6:65–70
77. Wilcoxon F (1945) Individual comparisons by ranking methods. Biometrics Bull 1:80–83
78. Sheoran K, Tomar P, Mishra R (2020) A novel quality prediction model for component based software system using ACO–NM optimized extreme learning machine. Cogn Neurodyn 14:509–522
79. Idri A, Hosni M, Abran A (2016) Systematic literature review of ensemble effort estimation. J Syst Softw 118:151–175