International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V72I2P102

Volume 72 Issue 2, 9-18, February 2024

Original Article

Iterative Feature Elimination Method Using Artificial Neural Network for Software Effort Estimation

Pranay Tandon[1], Ugrasen Suman[2]

[1,2] *School of Computer Science s& IT, Devi Ahilya University, Indore, M.P., India.*

[1]*Corresponding Author: pranay.tandon@live.com*

Received: 26 June 2023 Revised: 30 August 2023 Accepted: 06 January 2023 Published: 03 February 2024

Abstract *- Effort estimation is one of the critical tasks for any software development team because estimation is the key to planning the software development life cycle activities with proper timeline and cost. On-time and quality delivery is most important to build customer trust and certainty. There are many features to be considered while estimating the efforts, but removing the weak features and finding the set of the strongest features for any estimation process is difficult. Deep learning is the most popular prediction technique for effort estimation because of its capacity to adapt and be accurate on different types of datasets. Artificial Neural Network is best suited to deep learning techniques for predicting effort, per industrial research. In this paper, a novel model based on artificial neural networks and an iterative feature elimination-based method has been proposed to estimate the efforts. With ranking features, the proposed method can find the optimized set of features to be used in the model and final efforts. COCOMO NASA 2 dataset is used to find the results.*

Keywords *- Iterative feature elimination, Artificial Neural Network, Software effort estimation, Machine Learning, Deep learning.*

1. Introduction

Estimation is one of the most critical activities of project management. For many years, Information technology professionals have faced problems with accurately estimating the effort, cost, and time required to develop any piece of work. Forecasting all the required parameters in the very initial stage of the software development life cycle is very challenging when boundaries of all requirements need to be established and when unpredictability regarding the functionalities of the final product is substantial.

Mostly, limited knowledge of influencing factors, associated risks, and legacy software estimation techniques may lead to imprecise and inaccurate estimates; as a result, they may severely impact project delivery schedule, budget, and quality [11] where a better estimation leads to efficient project planning, better resource management, on-time delivery, improved client relationship, standard quality of product and strong reputation of organization.

Many researchers and professionals have worked on this problem of software estimation, from expert judgment planning poker to Machine Learning (ML) techniques that have been explored and innovated. Deep learning, a part of advanced ML, is a very popular and modern technique for prediction based on data.

Artificial neural network (ANN) works on deep learning concepts to find the value of a dependent variable with the help of other variables' values. ANN works with features and the weightage of each feature in a multi-layer perceptron architecture, where many layers are included, such as the input layer, hidden layer, and output layer, as the features perform the most important role in ANN model; that is why the selection of features is very critical and key activity.An optimal set of features may lead to accurate results, time-saving, memory-saving, simple processes, and tuned models.

There are many features of the project to be considered while estimating the efforts, but removing the weak features and finding the set of the strongest features for any estimation process is difficult. The main problem being rectified in this research is the difficulty in selecting the most significant features. The objective of this research is to find a robust method to get the set of most significant features that perform a vital role in any prediction process and obtain the better accuracy of a model.

In the proposed method, the weakest features are eliminated from the set of all features, and an optimized set of features is found as a result of the overall method. An ANN is used to develop a classifier model. In each phase of this proposed method, an ANN classifier is used with a different set of features per the algorithm's requirements.This paper is organized as follows: Section 2 describes some important literatures and research on software effort estimation. Section 3 briefly explained the methodology, including the Iterative feature elimination method and the ANN model as the classifier. Section 4 describes the proposed method with the flow diagram and algorithm. Section 5 outlines the setup and analysis part of the experiment.

Section 6 presents the experiment result and discussion of the result of the proposed method compared to other related work. Finally, Section 7 concluded the work with possible future directions.

2. Literature Review

As we are focusing mainly on ML techniques for effort estimation, many software effort estimation approaches have been used till now, such as Planning Poker (PP), function points, expert judgment, analogy, disaggregation, and algorithmic approach. Nowadays, ML is a new and popular tool for effort estimation.

In recent years, ML-based methods have received increasing attention in software development effort estimation research. Many ML techniques have been reviewed in this literature survey. A Decision Tree (DT) is used with PP, and it is found that PP with DT and PP with a logistic model tree are better than PP alone. Multiple ML algorithms or ensemble-based algorithms can be used with PP [1].

Linear regression and K-nearest neighbours ML techniques were compared, and from the results Linear Regression model is found better estimator than K-nearest neighbours on the data sets COCOMO81, COCOMO NASA, COCOMO NASA 2 by having higher correlation coefficient value and low Relative Absolute Error (RAE), Root Relative Squared Error (RRSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) [16].

Naïve Bayes (NB), Logistic Regression (LR), and Random Forest (RF) are explored and compared. In conclusion, RF obtained the best performance among these. This research can be extended with data mining ML methods [2]. Term Frequency - Inverse Document Frequency (TF-IDF) and doc2vec text vectorization are used with Gaussian Naive Bayes and SVM ML methods and found that better estimations can be obtained than COCOMO. Large data sets may be used in future [3]. Different ML techniques such as DT, RF, and Stochastic Gradient Boosting (SGB) are used with the Story Point Approach (SPA), and compared, SGB is found to be the best among them. Limitations are the small size data set and assumptions of the initial project velocity value of the team. Further extreme learning machines and Bayesian Networks (BN) on the SPA-related dataset can be used [4].The BN model is explored for estimation with more accuracy than other ML techniques. The proposed model is relatively simple and small; all the input data are easily evoked, so the impact on agility is minimal. The model can predict the efforts of a task, and it is independent of the agile methods used. It is also suitable to be used in the early project phase, but all influencing factors were not utilized in this approach [17].

In other research, the limitation of the BN model is the validation for future research; it validates the model in two stages: node probability tables validation and model validation. It can define more scenarios and can be compared in collaboration with experts [5]. Evolutionary Cost Sensitive Deep Belief Network (ECS-DBN) is introduced. This ECS-DBN model is relatively concise and uncomplicated, and all the input data are easily evoked. The scope of application can be increased to other deep learning techniques with higher dimensional data for better performance [6].ANN-feedforward back-propagation neural network, cascade-correlation neural network, and Elman neural network are compared. The Feedforward back-propagation network calculated better Effort estimation than 2 others. In the future, more ANNs can be compared using a large enough data set [7]. Long Short Term Memory (LSTM) is applied with regression as an activation function and recurrent highway network. This approach performed better than the existing TF-IDF technique in estimating the story points. The data set is small; this is the main limitation of this research. The feature selection process can be improved in the future [8].

Systematic Literature Review (SLR) is performed on ML methodologies, Ordinary least squares regression, selection operator regression, Ridge regression least absolute shrinkage, elastic-net regression, least angle regression, classification and regression tree, Analogy-Based Estimation (ABE), Support Vector Regression (SVR), adaptive boosting, RF, deep neural networks, ANN, bootstrap aggregating, and gradient boosting machine. Ensemble learning algorithms based on the principle of bootstrap aggregating, for example, Bagging and RF, performed the best overall over the 13 datasets. ABE appeared to be the highest-performing non-ensemble learning algorithm [9].

Ensemble-based model is also explored, and an application of this ensemble-based predictive model is developed. The ensemble-based prediction method is proven to be better than other prediction approaches. This approach is limited to the dataset from a particular organization, and some predictive algorithms in the ensemble provided better prediction results than this ensemble algorithm. For improvement, it can include human experts in ensembles and consider developing efficient optimization techniques at the project level [10].Another study used the ranking of features with Recursive Feature Elimination (RFE) and cross-validated selection of the feature numbers with the RF Classifier. The RF tree structure is used as the elimination classifier.

The dataset is reduced by 95% compared to the original size. The deep learning - DMLP model develops a smaller and more meaningful dataset by achieving an accuracy of 89% [12].In another experimental study, authors have proposed LR-RFE with a cross-validation-based feature selection method for classification. To avoid the overfitting problem with RFE, 10-fold stratified cross-fold validation is applied. After including top-ranked features, the pre-processed dataset is then applied to different ML models; LR performs best on all model evaluation measures used. It was also observed that the feature selection method on the few dimensions (8 independent features) has contributed to improving the model accuracy and has helped to avoid severe concerns like multicollinearity. [13].

In another proposed paper, an intrusion detection technique has been implemented that has been trained and tested on three ML classifiers, i.e., SVM, RF, and DT. In-depth research on the ML classifiers for all the features has

been conducted, and it has been found that some features are irrelevant and redundant in the dataset. Hence, RFE is used to reduce the dimensionality of the dataset. After comparing all three classification techniques, RF proved to perform better than SVM before feature selection. However, after implementing feature selection, SVM performed better than RF and DT. [14].

An effective method of software effort estimation based on RFE has been evolved, and the model has been tested with seven ML-based classifiers. From the output of this method, the ranking of features is decided, and the best features are passed as input to another model based on ensemble-based learning. In the proposed ensemble-based learning process, all seven methods are taken part and predicted the actual cost and Lines Of Code (LOC). Simulation results prove that out of the fifteen features considered, four features, database size, required software reliability, process complexity, and main memory constraint, are the least significant for both targets. The performance of this method in terms of both targets LOC and actual cost is quite encouraging compared to the individual ML methods [15].

As per the literature survey, there are some limitations in previous research, such as all the influencing factors and features were not selected, features were not optimized in a proper set of most impacting features, most of the algorithms were not giving a properly optimized set as final output rather giving just a way to select them with the manual intervention of an expert.As a solution, to resolve the problem of feature selection and overcome all the limitations of the literature survey, an automated iterative feature elimination method with a final optimized set of features as output can be evolved.

3. Methodology

The methodology includes the artificial neural network classification model, the iterative feature elimination method for finding rank, and an iterative set of feature elimination methods to find an optimized set. The flow of methodology is shown in Figure 1. In the proposed method, the ANN model is used for classification, and ANN is one of the most used supervised model functions, which can be seen as a multi-layer network of neurons. An ANN classification model is developed with 3 layers: input layer, hidden layer, and output layer. The model is trained and tested with different features of the data set in each iteration, and accuracy is found; all hyperparameters in this ANN model are used per output classes, dataset type, and size. The model is tuned for best performance.

Training and testing the ANN model with the set of optimized features is considered a critical task, as selecting an optimized set of features from any dataset is a complicated process. The iterative feature elimination method is the way to efficiently select the required set of features to train and test the ANN classification model. This method is used to eliminate the weakest features per rank from weaker to stronger and find the set of the features to be used finally in the model. The model's accuracy with all the features is compared to the model's accuracy after eliminating one feature, and this process continues for each feature to find the rank of features. After finding the rank of features from weaker to stronger, the iterative set of feature elimination method is used to remove the weakest set of features from a set of all features to find an optimized set of features to be used in the ANN model. This final set is used to train and test the resultant model, ready for effort estimation.

The novelty of the proposed method is a multilevel iteration process in an ANN prediction model. The first level finds the rank of the weakest features, and the second level finds the rank of the weakest set of features to be finally eliminated from the dataset. This method actually fine-tunes the performance of the model by selecting only the most eligible features for the model.

4. Iterative Feature Elimination Method with Classification Model

The proposed method consists of an Iterative feature elimination method and an ANN classification model. Accuracy is computed after each elimination of feature iteratively with the help of the ANN classification model, the rank of all features is decided, and then accuracy is computed after each elimination of a set feature iteratively as per the feature's rank with the help of the ANN classification model. Finally, the optimized set of features is found. Features and targets of the COCOMO NASA 2 dataset are shown in Table 1 and Table 2, respectively. 21 features and 2 targets are there in the dataset.

Table 1. COCOMO NASA 2 dataset features

Column Number	Attribute Name
0	project name
1	category of application
2	flight or ground system
3	NASA centre
4	year of development
5	development mode
6	Database size
7	Process complexity
8	Required software reliability
9	Time constraint for CPU
10	main memory constraint
11	machine volatility
12	turnaround time
13	analysts capability
14	application experience
15	programmers capability
16	virtual machine experience
17	language experience
18	modern programming practices
19	use of software tools
20	schedule constraint

Table 2. COCOMO NASA 2 Dataset Targets

Column Number	Attribute Name
21	LOC
22	Actual Efforts

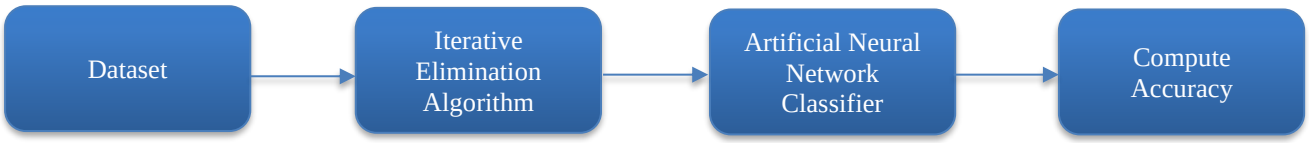

Fig. 1 Methodology

An ANN model for classification is developed with 3 layers: input layer, hidden layer, and output layer. The RELU activation function is used in layers, and the negative log-likelihood function is used as a loss function. The dataset was split into 65% for training and 35% for model testing.

The accuracy of the ANN classifier is observed separately for all twenty-one features with respect to LOC and actual cost target.In various steps of the proposed algorithm, this classifier is trained and tested with different inputs to find accuracy for ranking the features and set of features, followed by finding the final optimized set. Once the final optimized set is found, this classifier is used again for prediction after training and testing.

In the first step, the features affecting LOC and Actual cost in software effort estimation are ranked as per accuracy obtained by the ANN model for each feature eliminated. COCOMO Nasa 2 dataset has been considered in this work, having twenty-one number of features required, such as project name, category of application, flight or ground system, NASA centre, year of development, development mode, Database size, Process complexity, Required software reliability, a Time constraint for CPU, main memory constraint, machine volatility, turnaround time, analysts capability, application experience, programmers capability, virtual machine experience, language experience, modern programming practices, use of software tools and schedule constraint, and with two number of class labels such as LOC and actual cost.

This data set has been divided into two sets. In the first set, twenty-one features have been considered with the target LOC and others with the target actual cost. Both sets of datasets are being processed through the iterative elimination of features algorithm shown in Algorithm 1 and flow shown in Figure 2, where a single feature is eliminated at a time, and the remaining twenty features with LOC class label are input to the ANN classifier.

After building the classifier model with all the dataset features, the model is trained and tested, and found the accuracy A. In an iterative fashion, each feature is eliminated from the dataset, and found the accuracy A' with the remaining features in the dataset. The percentage difference in A' against A is considered for the ranking of features. After completing all the features using the above process, the rank of all features is found.

Algorithm 1: Ranking of features
Input : Dataset S = {f_1, f_2, …f_n}
Output : rank of features RF = {rf_1, rf_2, …rf_n}

1. *Build a classifier using dataset S*
2. *Prepare train data from dataset S and then train the classifier*
3. *Prepare test data from dataset S and then test the classifier*
4. *Find the accuracy of classifier for dataset S*
5. *For each feature f_i in S*
    1. *$S' = S - f_i$*
    2. *Prepare train data from dataset S' and then train the classifier*
    3. *Prepare test data from dataset S' and then test the classifier*
    4. *Find the accuracy of classifier for dataset S'*
    5. *Find the accuracy loss of classifier due to elimination of f_i*
6. *Create the accuracy loss set of features {f_1, f_2, …f_n}*
7. *Sort the set to find rank of features RF = {rf_1, rf_2, …rf_n}*

After the first step, the rank of features from weakest to strongest is found, which will be inputted in this second step. Starting from the weakest feature, the set of features is created by adding features one by one as per ranking. This set of features is eliminated from all the features in the dataset and found the accuracy of classifier after training and testing. This process continues until all the input set is iterated.

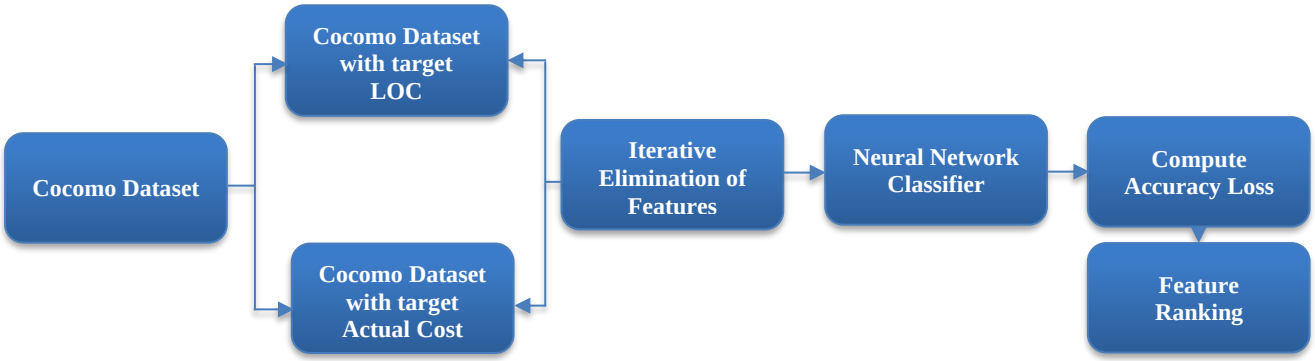

Fig. 2 Ranking of feature

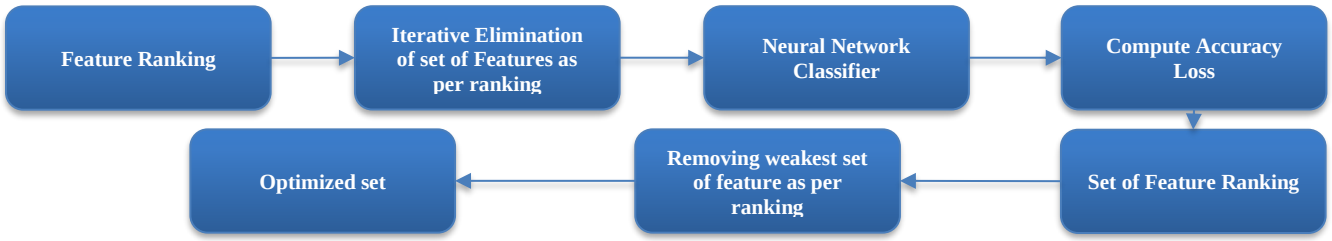

Fig. 3 Ranking of the set of features and finding an optimized set

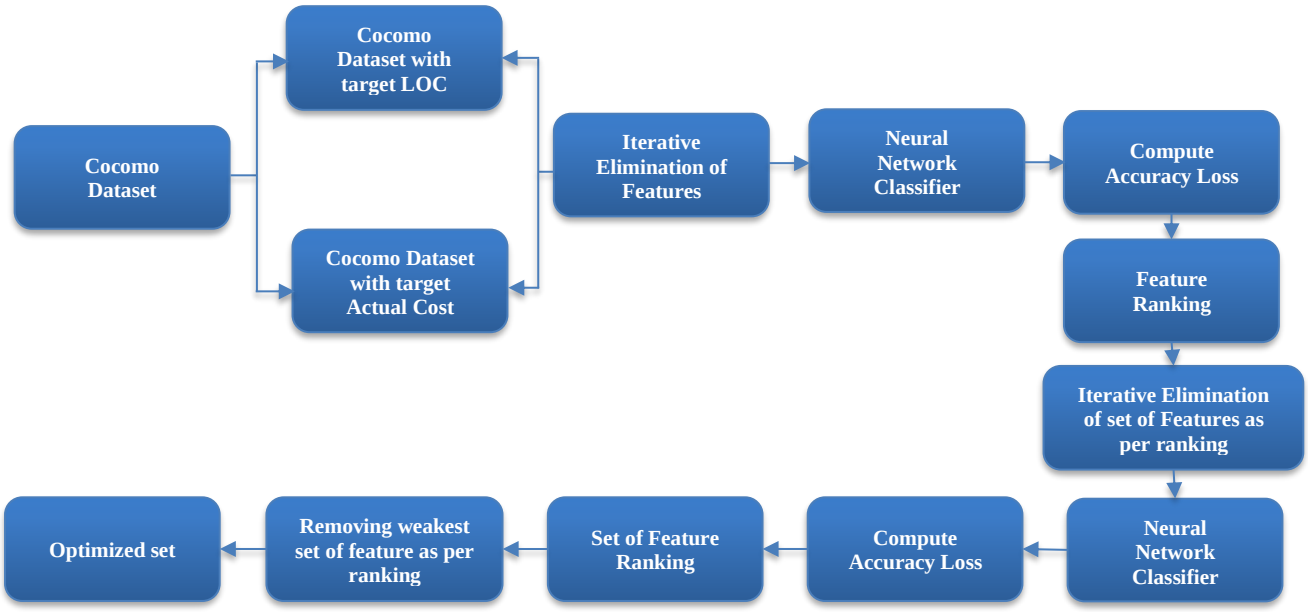

Fig. 4 Iterative feature elimination method and finding an optimized set

The rank of the set of features is found, the weakest set of features is removed from all the features in the dataset, and finally, the optimized set is found. The flow is presented in Figure 3, and the algorithm is shown in Algorithm 2. The full flow combining all steps with two algorithms and one classification model is depicted in Figure 4 below, and a full algorithm is also described in Algorithm 3.

Algorithm 2: Ranking of a set of features and finding an optimized set

Input : rank of features RF = {rf_1, rf_2, …rf_n}

Output : Optimized set of features OS = {f_1, f_2, …f_n}

1. *For each feature in set RF*
    1. *Removed features set $ReF_i = ReF_{i-1} + rf_i$*
    2. *$S' = S - ReF_i$*
    3. *Prepare train data from dataset S' and then train the classifier*
    4. *Prepare test data from dataset S' and then test the classifier*
    5. *Find the accuracy of classifier for dataset S'*
    6. *Find the accuracy loss of classifier due to elimination of ReF_i*
2. *Create the accuracy loss set of features set {ReF_1, ReF_2, …ReF_n}*
3. *Sort the set to find rank of features set RReF = {$rReF_1$, $rReF_2$, …$rReF_n$}*
4. *Find removed features set with the most accuracy RReF*
5. *Finally, find the optimized set OS = S – RReF*

Algorithm 3 Iterative feature elimination method and finding optimized set

Input : Dataset S = {f_1, f_2, …f_n}

Output : Optimized set of features OS = {f_1, f_2, …f_n}

1. *Build a classifier using dataset S*
2. *Prepare train data from dataset S and then train the classifier*
3. *Prepare test data from dataset S and then test the classifier*
4. *Find the accuracy of classifier for dataset S*
5. *For each feature f_i in S*
    1. *$S' = S - f_i$*
    2. *Prepare train data from dataset S' and then train the classifier*
    3. *Prepare test data from dataset S' and then test the classifier*
    4. *Find the accuracy of classifier for dataset S'*
    5. *Find the accuracy loss of classifier due to elimination of f_i*
6. *Create the accuracy loss set of features {f_1, f_2, …f_n}*
7. *Sort the set to find rank of features RF = {rf_1, rf_2, …rf_n}*
8. *For each feature in set RF*
    1. *Removed features set $ReF_i = ReF_{i-1} + rf_i$*
    2. *$S' = S - ReF_i$*
    3. *Prepare train data from dataset S' and then train the classifier*
    4. *Prepare test data from dataset S' and then test the classifier*

5. *Find the accuracy of classifier for dataset S'*
6. *Find the accuracy loss of classifier due to elimination of* ReF_i

9. *Create the accuracy loss set of features set* $\{ReF_1, ReF_2, \ldots ReF_n\}$
10. *Sort the set to find rank of features set* $RReF = \{rReF_1, rReF_2, \ldots rReF_n\}$
11. *Find removed features set with the most accuracy RReF*
12. *Finally, find the optimized set OS = S – RReF*

5. Experimental Setup and Analysis

This experiment was performed using a system having Intel(R) Core(TM) i5-8250U CPU @1.60 GHz, 1800 MHz, 4 Core(s), 8 Logical Processor(s), 8.00 GB RAM, and 64-bit OS Windows 10 configurations. The simulation environment includes Java JDK 11, deeplearning4j and nd4j API, and Eclipse 2022 -12 IDE. All hyperparameters of the classifier are set by selecting suitable values on a trial-and-error basis.

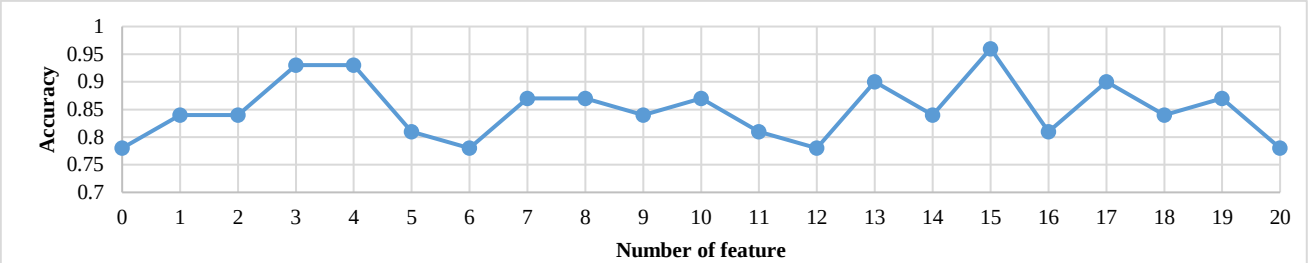

Fig. 5 Accuracy for target LOC after one feature elimination

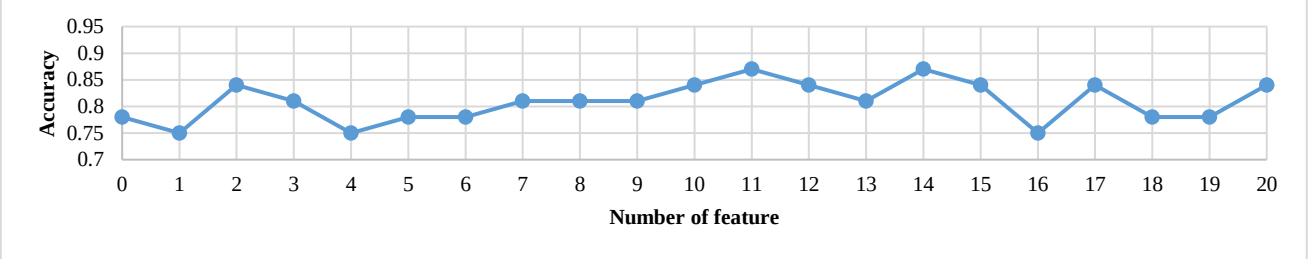

Fig. 6 Accuracy for target actual cost after one feature elimination

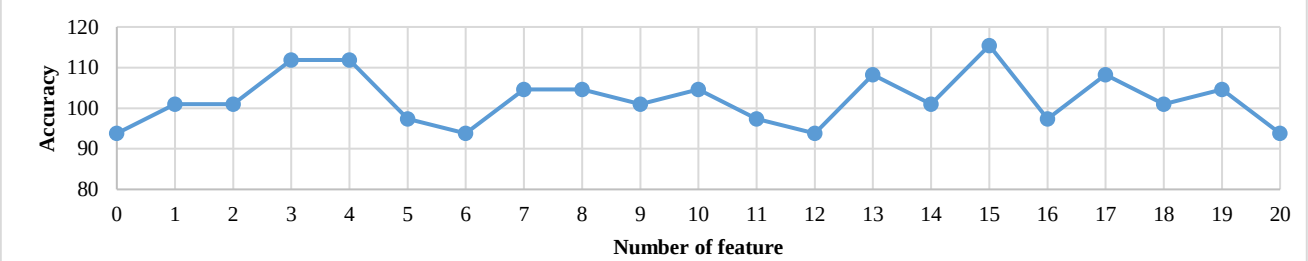

Fig. 7 Accuracy percent for target LOC after one feature elimination against all features

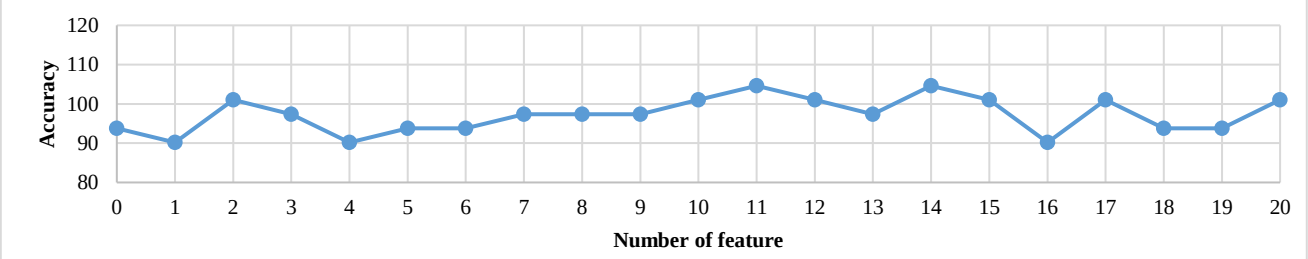

Fig. 8 Accuracy percent for target actual cost after one feature elimination against all features

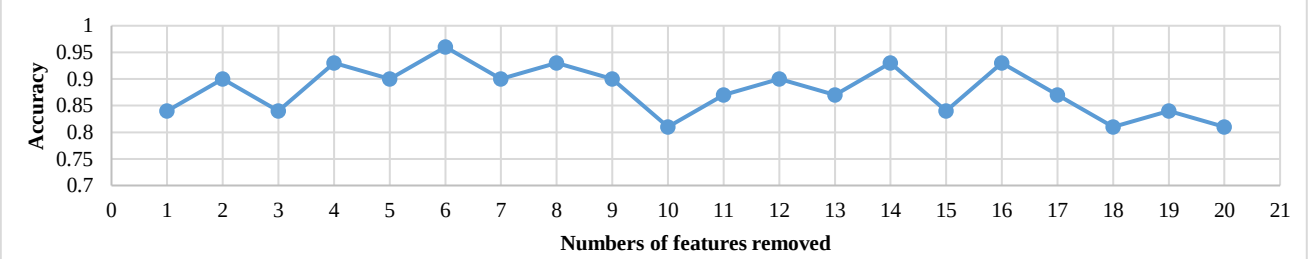

Fig. 9 Set of features and Accuracy for target LOC

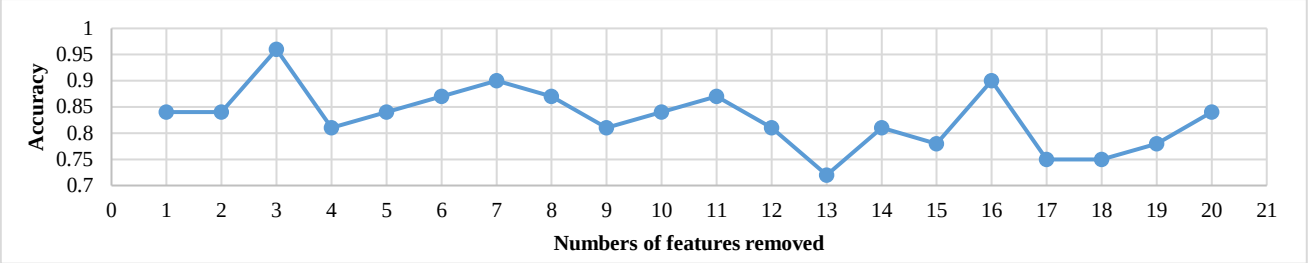

Fig. 10 Set of features and Accuracy for target actual cost

Figure 5 presents the accuracy of the ANN classifier for the target LOC after eliminating one feature. Here, after removing feature number 15, the accuracy of the classifier is 0.96, which has the most accuracy. After removing the feature numbers 0, 6, 12, and 20, the accuracy of the classifier is 0.78, which is the lowest accuracy.

Figure 6 presents the accuracy of the ANN classifier for the target actual cost after eliminating one feature. Here, after removing feature numbers 11 and 14, the accuracy of the classifier is 0.87, which has the most accuracy. After removing the feature numbers 1, 4, and 16, the accuracy of the classifier is 0.75, which is the lowest accuracy.

Figure 7 presents the percentage of the classifier's accuracy for target LOC increased or decreased after eliminating one feature against the classifier's accuracy of all features.

Figure 8 presents the percentage of the classifier's accuracy for target actual cost increased or decreased after eliminating one feature against the classifier's accuracy of all features. The accuracy of the ANN classifier using all the features of the COCOMO NASA 2 dataset is 0.84; the same accuracy is considered to find other values in the proposed method.Figure 9 presents the accuracy for target LOC after the number of features removed rank-wise from all feature sets. The below graph shows that the accuracy after removing the first 6 weakest features is 0.96, which is the most accuracy, and the accuracy after removing the first 10,18 and 20 weakest features is 0.81, which is the least accuracy.

Figure 10 presents the accuracy for the target actual cost after the number of features removed rank-wise from all feature sets. In the graph below shows that the accuracy after removing the first 3 weakest features is 0.96, which is the most accuracy, and the accuracy after removing the first 13 weakest features is 0.72, which is the least accuracy.

For target LOC, the accuracy of all the removed sets of columns is shown in Table 3. So the removed set includes the first 6 weakest features {15, 3, 4, 13, 17, 19} presented in Table 4; with the help of this, the optimized set of features to be used in the ANN model can be found in {0,1,2,5,6,7,8,9,10,11,12,14,16,18,20} which is described in Table 5. Similarly, for the target actual cost, the accuracy of all the removed sets of columns is shown in Table 6. So the removed set includes the first 3 weakest features {11,14,12} presented in Table 7; with the help of this, the optimized set of features to be used in the ANN model can be found in {0,1,2,3,4,5,6,7,8,9,10,13,15,16,17,18,19,20} which is described in Table 8.

Table 3. Removed columns and accuracy for target LOC

Removed columns	Accuracy
{15}	0.84
{15, 3}	0.9
{15, 3, 4}	0.84
{15, 3, 4, 13}	0.93
{15, 3, 4, 13, 17}	0.9
{15, 3, 4, 13, 17, 19}	0.96
{15, 3, 4, 13, 17, 19, 7}	0.9
{15, 3, 4, 13, 17, 19, 7, 8}	0.93
{15, 3, 4, 13, 17, 19, 7, 8, 10}	0.9
{15, 3, 4, 13, 17, 19, 7, 8, 10, 14}	0.81
{15, 3, 4, 13, 17, 19, 7, 8, 10, 14, 18}	0.87
{15, 3, 4, 13, 17, 19, 7, 8, 10, 14, 18, 1}	0.9
{15, 3, 4, 13, 17, 19, 7, 8, 10, 14, 18, 1, 2}	0.87
{15, 3, 4, 13, 17, 19, 7, 8, 10, 14, 18, 1, 2, 9}	0.93
{15, 3, 4, 13, 17, 19, 7, 8, 10, 14, 18, 1, 2, 9, 11}	0.84
{15, 3, 4, 13, 17, 19, 7, 8, 10, 14, 18, 1, 2, 9, 11, 16}	0.93
{15, 3, 4, 13, 17, 19, 7, 8, 10, 14, 18, 1, 2, 9, 11, 16, 5}	0.87
{15, 3, 4, 13, 17, 19, 7, 8, 10, 14, 18, 1, 2, 9, 11, 16, 5, 12}	0.81
{15, 3, 4, 13, 17, 19, 7, 8, 10, 14, 18, 1, 2, 9, 11, 16, 5, 12, 0}	0.84
{15, 3, 4, 13, 17, 19, 7, 8, 10, 14, 18, 1, 2, 9, 11, 16, 5, 12, 0, 6}	0.81

Table 4. Removed set of features for target LOC

Column Number	Attribute Name
3	NASA centre
4	year of development
13	analysts capability
15	programmers capability
17	language experience
19	use of software tools

Table 5. Optimized set of features for target LOC

Column Number	Attribute Name
0	project name
1	category of application
2	flight or ground system
5	development mode
6	Database size
7	Process complexity
8	Required software reliability
9	Time constraint for CPU
10	main memory constraint
11	machine volatility
12	turnaround time
14	application experience
16	virtual machine experience
18	modern programming practices
20	schedule constraint

Table 6. Removed columns and accuracy for target actual cost

Removed columns	Accuracy
{11}	0.84
{11, 14}	0.84
{11, 14, 12}	0.96
{11, 14, 12, 15}	0.81
{11, 14, 12, 15, 17}	0.84
{11, 14, 12, 15, 17, 2}	0.87
{11, 14, 12, 15, 17, 2, 20}	0.9
{11, 14, 12, 15, 17, 2, 20, 10}	0.87
{11, 14, 12, 15, 17, 2, 20, 10, 13}	0.81
{11, 14, 12, 15, 17, 2, 20, 10, 13, 3}	0.84
{11, 14, 12, 15, 17, 2, 20, 10, 13, 3, 7}	0.87
{11, 14, 12, 15, 17, 2, 20, 10, 13, 3, 7, 8}	0.81
{11, 14, 12, 15, 17, 2, 20, 10, 13, 3, 7, 8, 9}	0.72
{11, 14, 12, 15, 17, 2, 20, 10, 13, 3, 7, 8, 9, 18}	0.81
{11, 14, 12, 15, 17, 2, 20, 10, 13, 3, 7, 8, 9, 18, 19}	0.78
{11, 14, 12, 15, 17, 2, 20, 10, 13, 3, 7, 8, 9, 18, 19, 0}	0.9
{11, 14, 12, 15, 17, 2, 20, 10, 13, 3, 7, 8, 9, 18, 19, 0, 5}	0.75
{11, 14, 12, 15, 17, 2, 20, 10, 13, 3, 7, 8, 9, 18, 19, 0, 5, 6}	0.75
{11, 14, 12, 15, 17, 2, 20, 10, 13, 3, 7, 8, 9, 18, 19, 0, 5, 6, 16}	0.78
{11, 14, 12, 15, 17, 2, 20, 10, 13, 3, 7, 8, 9, 18, 19, 0, 5, 6, 16, 1}	0.84

Table 7. Removed set of features for target actual cost

Column Number	Attribute Name
11	machine volatility
12	turnaround time
14	application experience

Table 8 Optimized set of features for target actual cost

Column Number	Attribute Name
0	project name
1	category of application
2	flight or ground system
3	NASA centre
4	year of development
5	development mode
6	Database size
7	Process complexity
8	Required software reliability
9	Time constraint for CPU
10	main memory constraint
13	analysts capability
15	programmers capability
16	virtual machine experience
17	language experience
18	modern programming practices
19	use of software tools
20	schedule constraint

Table 9. Accuracy of classifiers after the feature elimination method applied

Classification model	Accuracy with target LOC	Accuracy with target actual cost
Decision tree	92.18	79.68
KNN	90.62	79.68
Logistic regression	67.18	70.31
Naïve Bayes	62.5	73.43
Random forest	92.18	73.43
Stochastic gradient descent	46.87	54.68
SVM	53.12	67.18
Ensemble model	95.31	81.25
The proposed ANN model	96.96	96.96

6. Result and Discussion

In the literature survey, the feature elimination method's use cases and results in estimations using ML techniques have been reviewed, and it was found that this proposed iterative feature elimination method with the ANN model for classification has outperformed other models.

This proposed model uses this iterative elimination twice in the algorithm to obtain the most optimized set of features. After applying the proposed method to features of the COCOMO NASA 2 data set, the model's classification accuracy increased up to 96% for both actual cost and LOC target. The accuracy of different classifier models is presented in Table 9. After comparison, it is found that the accuracy of the proposed ANN classification model with the iterative feature elimination method is the highest for both targets LOC and actual cost.

This proposed method is beneficial to information technology professionals because it allows them to find more accurate software effort estimation. Professionals can use deep learning with the optimum memory requirement, process time, and resources, as decreasing the number of features can improve the model's performance. Academicians can use this paper to train and educate students in this subject. Researchers can use this research to innovate new possibilities in this area further and add new research works by extending this.

7. Conclusion

Deep learning has always been a popular and efficient technique to predict software effort, cost, time, and size. Selecting features for a deep learning model is always the most important task to avoid overestimating or underestimating a piece of work in software development. If there is a significant method available for feature selection, then this task becomes easy and standardized.

This research proposes a novel method of software effort estimation based on Iterative feature elimination. The dataset is trained and tested with the ANN model for finding accuracy, and then the features are being ranked. The weakest features are eliminated from the set of all features iteratively, and the set of weakest features is found to be eliminated from all features dataset. The performance of the proposed iterative feature elimination method concerning LOC and actual cost is quite encouraging compared to the individual methods. For target LOC, the accuracy of individual ML technique Random forest and decision tree is 92.18%, and the accuracy of the ensemble model is 95.3%, where the accuracy of this proposed ANN model with iterative feature elimination method is more than 96%. Similarly, for target Actual cost accuracy of the Decision tree and KNN is 79.68 %, the ensemble model achieved more than 81%, and the proposed ANN model's accuracy is 96.96%As future work, the proposed method and model

may be used for other agile project datasets and hyperparameters of ANN can be tuned further to increase accuracy.

The strength of this proposed method is that the levels of iterative elimination are more than in the previous pieces of literature. The limitation is that it consists of a neural network as the main deep learning technique used for the model, and it requires more resources and a large and proper dataset with minimal missing data for any attributes. Sometimes, neural networks cannot justify the result because of its random nature.

Conflict of Interest

The authors declare that this research manuscript has no conflict of interest with any other published source and has not been published previously (partly or in full). No data have been fabricated or manipulated to support the conclusions.

References

[1] Kayhan Moharreri et al., "Cost-Effective Supervised Learning Models for Software Effort Estimation in Agile Environments," *IEEE 40th Annual Computer Software and Applications Conference*, Atlanta, GA, USA, pp. 135-140, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[2] Ahmed BaniMustafa, "Predicting Software Effort Estimation Using Machine Learning Techniques," *2018 8th International Conference on Computer Science and Information Technology*, Amman, Jordan, pp. 249-256, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[3] Vlad-Sebastian Ionescu, Horia Demian, and Istvan-Gergely Czibula, "Natural Language Processing and Machine Learning Methods for Software Development Effort Estimation," *Studies in Informatics and Control*, vol. 26, no. 2, pp. 219-228, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[4] Shashank Mouli Satapathy, and Santanu Kumar Rath, "Empirical Assessment of Machine Learning Models for Agile Software Development Effort Estimation Using Story Points," *Innovations in Systems and Software Engineering*, vol. 13, pp. 191-200, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[5] Laura-Diana Radu, "Effort Prediction in Agile Software Development with Bayesian Networks," *Proceedings of the 14th International Conference on Software Technologies*, Prague, Czech, vol. 1, pp. 238-245, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[6] Hosahalli Mahalingappa Premalatha, and Chimanahalli Venkateshavittalachar Srikrishna, "Effort Estimation in Agile Software Development Using Evolutionary Cost-Sensitive Deep Belief Network," *International Journal of Intelligent Engineering and Systems*, vol. 12, no. 2, pp. 261-269, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[7] Saurabh Bilgaiyan, Samaresh Mishra, and Madhabananda Das, "Effort Estimation in Agile Software Development Using Experimental Validation of Neural Network Models," *International Journal of Information Technology*, vol. 11, pp. 569-573, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[8] Morakot Choetkiertikul et al., "A Deep Learning Model for Estimating Story Points," *IEEE Transactions on Software Engineering*, vol. 45, no. 7, pp. 637-656, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[9] Passakorn Phannachitta, and Kenichi Matsumoto, "Model-Based Software Effort Estimation–A Robust Comparison of 14 Algorithms Widely Used in the Data Science Community," *International Journal of Innovative Computing, Information and Control*, vol. 15, no. 2, pp. 569-589, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[10] Onkar Malgonde, and Kaushal Chari, "An Ensemble-Based Model for Predicting Agile Software Development Effort," *Empirical Software Engineering*, vol. 24, pp. 1017-1055, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[11] Przemyslaw Pospieszny, Beata Czarnacka-Chrobot, and Andrzej Kobylinski, "An Effective Approach for Software Project Effort and Duration Estimation with Machine Learning Algorithms," *Journal of Systems and Software*, vol. 137, pp. 184-196, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[12] Serpil Ustebay, Zeynep Turgut, and Muhammed Ali Aydin, "Intrusion Detection System with Recursive Feature Elimination by Using Random Forest and Deep Learning Classifier," *International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism*, Ankara, Turkey, pp. 71-76, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[13] Puneet Misra, and Arun Singh Yadav, "Improving the Classification Accuracy Using Recursive Feature Elimination with Cross-Validation," *International Journal on Emerging Technologies*, vol. 11, no. 3, pp. 659-665, 2020. [Google Scholar] [Publisher Link]

[14] Neha V. Sharma, and Narendra Singh Yadav, "An Optimal Intrusion Detection System Using Recursive Feature Elimination and Ensemble of Classifiers," *Microprocessors and Microsystems*, vol. 85, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[15] K. Eswara Rao, and G. Appa Rao, "Ensemble Learning with Recursive Feature Elimination Integrated Software Effort Estimation: A Novel Approach," *Evolutionary Intelligence*, vol. 14, no. 1, pp. 151-162, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[16] Bhaskar Marapelli, "Software Development Effort Duration and Cost Estimation Using Linear Regression and K-Nearest Neighbors Machine Learning Algorithms," *International Journal of Innovative Technology and Exploring Engineering*, vol. 9, no. 2, pp. 1043-1047, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[17] Srdjana Dragicevic, Stipe Celar, and Mili Turic, "Bayesian Network Model for Task Effort Estimation in Agile Software Development," *Journal of Systems and Software*, vol. 127, pp. 109-119, 2017. [CrossRef] [Google Scholar] [Publisher Link]