Learning Automata based Feature Selection for Duplex Output Software Effort Estimation Model

Kapil Ravi Rathod
Department of Computer Science and Engineering
NIT Warangal, India
kapilravi@student.nitw.ac.in

Shruti Bansal
Department of Computer Science and Engineering
NIT Warangal, India
sbansal@student.nitw.ac.in

Yash Harish Chandra Kandpal
Department of Computer Science and Engineering
NIT Warangal, India
kyashharishchandra@student.nitw.ac.in

Manjubala Bisi
Department of Computer Science and Engineering
NIT Warangal, India
manjubalabisi@nitw.ac.in

2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT) | 978-1-7281-8595-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCCNT51525.2021.9579968

***Abstract*—Software Effort Estimation (SEE) plays an instrumental role in estimating the effort required to complete a software development project. It involves the process of estimating the effort required in order to meet project delivery deadlines by appropriately scheduling and allocating resources during the Software Development Life Cycle (SDLC). In this paper, we have proposed a feature selection technique with regression models using learning automata to estimate software development effort. We have validated our approach in six data sets for software effort estimation. Further, we have discretized the predicted effort into three different classes : High (H), Moderate (M) and Low (L). We have used various regression models including Linear, LASSO , ElasticNet and Ridge regression to obtain the estimated effort and use quartiles for discretizing them. We have compared the result of our proposed approach with some existing models of feature selection and regression techniques and found that our approach is able to provide better prediction performance than some existing models.**

***Index Terms*—Software development effort, Learning automata, Duplex output, Regression model, Optimization.**

I. Introduction

Software Effort Estimation (SEE) is the process of estimating the effort needed for executing a project in order to set up effective project delivery deadlines [1]. Accurately estimating the software effort of a new project is instrumental in the field of software engineering, since it minimizes over and underestimation, both of which, may result in the cancellation of a project due to overruns in budget and can delay project funding. According to [1], no benchmark exists to validate estimated effort of a software project. In most cases, estimations by existing SEE models contradict the true/actual efforts on various validation sets due to inconsistencies within the data sets. These are some of the major challenges faced by the SEE research community and one of them is to further discretize the predicted effort into effort classes namely High, Moderate or Low effort.

Hence, accurate SEE allows for effective scheduling and management of resources allocated to various software projects. In this work, we have proposed a learning automata based feature selection technique [2] which is applied before predicting and discretizing the effort into 3 broad effort classes using quartiles. Learning automata is a reinforcement learning based technique which possesses good optimization characteristics and is a good fit to apply to feature selection [3]. Feature optimization is of paramount importance in machine learning and predictive tasks/applications as it reduces the number of dimensions of high dimensional data and improves the efficiency of an algorithm by reducing computational costs.

The major contributions of this work are as follows:

- Design of a regression model to predict software development effort in two dimensional output. Output 1 is the software development effort and Output 2 is software development effort classification.
- Development of a learning automata based approach to select the features affecting software development effort.
- Validation of the proposed approach on six data sets and comparison of results with some existing methods.

The rest of this paper is organized as follows: Related work to this paper are discussed in Section II. Our proposed approach is discussed in Section III.The results of our experiments are discussed in Section IV and a conclusion is drawn in Section V.

II. Related Work

There is no state-of-the-art model for software effort estimation which gives better predictions for all the datasets for a simple reason being that the existing models have their own flaws; one model may have high accuracy for estimation for a certain data set, and very low accuracy for another data set.

The number of features which are required for software development effort estimation were determined using different optimization algorithms [1]. One of the earliest and the most well documented model for software effort estimation was the COCOMO model. Constructive Cost Model (COCOMO) was proposed in [4] to estimate software development effort. Its parameters were derived by fitting a regression model using data from historical projects.

A baseline SEE model called ATLM (Automatically Transformed Linear Model) was proposed in [5] to estimate software effort. In this model, common regression models such as LASSO [6], ElasticNet [7], Ridge [8] and Robust [9] regression models were benchmarked against the ATLM base model to produce state-of-the-art results.In [10], a study was conducted to investigate whether software development effort estimation model (SSE) providing better prediction accuracy than some traditional SEE models.It was observed that the traditional model proposed in [4] had better accuracy than the new models.

In [11], a deep learning based model was used to investigate the use of prior and posterior features in SEE model. Multiple prediction models were investigated in [12] to estimate the conclusion instability issue from different data sets. In a study conducted by [13], the moving window concept was compared with the growing portfolio concept where historical software projects were used as the training set. A study conducted by [14] showed that the ordinary least square regression technique was the most effective software development effort prediction over nine datasets. Artificial Bee Colony Optimization method was integrated with analog- based estimation mode to predict software development effort in [15]. Most of the studies in literature had predicted software development effort in numeric value only. A few works have reported software development effort in duplex form i.e prediction of development effort as well as development class.

In this paper, we have used learning automata to select the most desirable features to predict software development effort in duplex form.. We have used different regression models with learning automata to predict software development in dual form i.e amount of effort and effort class.

III. Proposed Approach

The proposed approach to predict software development effort is shown in Figure 1. Learning automata (LA) is used to select the appropriate features and the duplex output model is developed using four regression models. The concept of LA is presented in Section III-A. The four regression models used for duplex software effort prediction are briefly discussed in Section III-B and proposed algorithms are shown in Section III-C.

A. Learning Automata

Learning Automata(LA) is a reinforcement learning based technique. It learns from a stochastic random environment by constantly interacting with it so as to maximize long run gains. The LA based architecture consists of 4 basic building blocks, namely: feedback, action, the learning automata itself and the stochastic environment.

At every time instant *t*, a particular action $\alpha(t)$ is selected by LA from a set called the Action Set. It is a set of all possible actions the LA can chose and once an action is chosen, it interacts with the environment.

When an action is chosen,the stochastic environment will supply a feedback $\beta(t)$ to the LA and based on the feedback returned, the environmental adaptability of the chosen action is hence reflected. After some iterations,the LA will start converging to an optimal value/action.

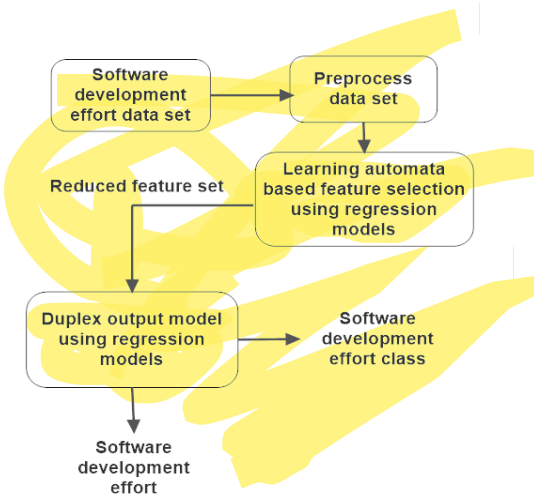

Fig. 1. **Proposed Approach**

B. Regression Models

In this paper, we use four regression models namely - Linear Regression, LASSO Regression, ElasticNet Regression and Ridge Regression. In this section we briefly describe each of the models used.

Linear Regression: Linear regression is used for the purpose of estimation of software effort and is hence used to predict the relationship between independent variables(X)(software metrics in our case) and a single dependent variable(y)(the predicted effort). Here the independent variables(X) can be two or more than two in number.

y is called the dependent variable because its value depends on the values taken by the independent variables(X).

The relationship is depicted below:

$$y = \beta_0 + \beta_1 X_1 + ... + \beta_n X_n$$

The dependent variable is y and the independent variables are X_1-X_n.

Here β_i depicts the regression coefficient of the *ith* independent variable X_i.

Ridge Regression:

Ridge regression [8] is essentially an extension of simple linear regression. It makes the use of the L2 regularization technique to prevent over fitting.

Ridge regression has a penalty factor λ which penalizes a higher values of the regression coefficients.

The optimal regression coefficients for Ridge regression are shown in Eq. 1.

$$\hat{\beta}^{\text{ridge}} = \arg\min_{\beta} \sum_{n=1}^{N} \frac{1}{2} (y_n - \beta x_n)^2 + \lambda \sum_{i=1}^{p} \beta_i^2 \qquad (1)$$

where N is the number of samples in the dataset.

In the vectorized notation, Eq. 1 can be written as:

$$\hat{\beta}_{\text{ridge}} = (X'X + \lambda I_p)^{-1} X'Y \quad (2)$$

where X'X is the singleton matrix of predictive variables and I_p is the identity matrix of rank p.

LASSO Regression:

Least Absolute Shrinkage and Selection Operator regression (LASSO) [6] is a regularization technique. This regression model makes use of the concept of shrinkage wherein the data values are made to shrink towards a central mean point. LASSO regression makes use of the L1 regularization technique to avoid overfitting. The optimal regression coefficients for LASSO regression are given in Eq. 3 below.

$$\hat{\beta}^{\text{lasso}} = \arg\min_{\beta} \sum_{n=1}^{N} \frac{1}{2} (y_n - \beta x_n)^2 + \lambda \sum_{i=1}^{p} |\beta_i| \quad (3)$$

where, N is the total number of samples in the data set.

ElasticNet Regression : ElasticNet regression [7] is nothing but a combination of both LASSO and Ridge regression techiques. It makes use of the L1 and L2 regularization techniques.

The optimal regression coefficients for ElasticNet regression are shown below in Eq. 4 .

$$\hat{\beta}^{\text{elastic}} = \underset{\beta}{argmin} \left\{ \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2 \right\} \quad (4)$$

where N is the number of samples in the dataset.

Here, λ_1 and λ_2 are the penalty constants used in L1 and L2 regularization respectively.

C. Algorithm

In our proposed method, we have combined the stability of the LA based feature selection algorithm [2] and the robustness of the Duplex Software Estimation algorithm [1]. We have proposed an improved algorithm as Improved Duplex Output Software Effort Estimation(IDOSEE), for efficiently predicting and discretizing the software effort required for a project. In our proposed method, we have used **Algorithm 1(LAFS)** as a subroutine in **Algorithm 2(IDOSEE)** which is an improved version of the Duplex Output Software Effort Estimation proposed in [1].

Before running **Algorithm 1(LAFS)**, we have divided the features of the input dataset into feature categories, wherein each feature category represents a single learning automata. For every feature category/learning automata, we have initialized the action probability vectors as 1/*r*, where the number of features in each category is *r*. Once the action probability vectors are initialized, we appropriately set the other input parameters namely,T_1(Accuracy Threshold) T_2(Action Probability Threshold) and Q(On-Off Probability).

Following are the values we have set to the above mentioned parameters:

TABLE I
PERCENTAGE VALUES SET FOR T_1,T_2 AND Q

Parameter	Value Set
T_1	70
T_2	70
Q	50

LA based Feature Selection (LAFS) The working principle of the LA-based Feature Selection (LAFS) is shown in Algorithm 1. LA is used to select the feature subset from the feature set. The improved duplex output software effort estimation (IDOSEE) algorithm is shown in Algorithm 2

Algorithm 1 LA based Feature Selection (LAFS)

procedure LAFS (T_1, T_2, P, R, Δ, Q, A)
  while *atleast one feature is present in every LA*
  Determine the ON-OFF state for each LA as per the LA ON-OFF probability Q at any instance of time *t*.
  Select an action $\alpha(t) = \alpha_i$ using the action probability vector P(*t*) of each ON state LA.
  Update the action set A
    if A=∅
      Go to Step 3.
  Determine the training instances of the data reserving the features present in R.
  Select randomly two mini-batch sets i.e. T(*t*) and V(*t*) from the training data, such that T(*t*) ≠ V(*t*).
  Remove all features present in the action set A from T(*t*) and V(*t*).
  Train and test the regression model using T(t) and V(t).
  Calculate the accuracy of the model at a time instant *t* as Accuracy(*t*).
  if Accuracy(*t*) ≥ Accuracy Threshold T_1
    Reward the current action set A with feedback as $\beta(t) = 0$.
  Update the probability vector P for ON LA according to the following update rule:
    $p_j(t+1) = \max\{ p_j(t) - \Delta, 0\}$, $\forall\, j \neq i$
    $p_i(t+1) = \min\{ 1 - \sum_{j \neq i} p_j(t), 1\}$
  if max(P(*t*)) ≥ Action Probability Threshold T_2
    Determine the maximum action probability value having index m.
    Discard the m^{th} feature from R.
  end while
  return R
end procedure

The most efficient features are selected using Algorithm 1 using LA. Depending upon the accuracy obtained by regression models, feedback is given to LA to update its probability

Algorithm 2 Improved Duplex Output Software Effort Estimation (IDOSEE)

procedure IDOSEE (X, Y, Q_L, Q_M, Q_H)
  Import Dataset.
  Preprocess the Dataset.
  FEATURE SELECTION USING LAFS
  for i = 1,. . ., q :
    Selecting X[r] ← X[1] to X[i]
  end for
CONSTRUCTING THE REGRESSION MODEL
  Applying the Min-Max Normalization on Dataset
  for j = 1,. . ., N :
    Performing Leave-One-Out(LOO) Validation
  end for
  for k = 1,. . ., r :
    Calculate $\beta_r \leftarrow$ inv(X'X)X'Y
  end for
  $Y_R \leftarrow \beta_0 + \Sigma\ \beta_i X_i$
  Y_R - Predicted Effort
  EFFORT CLASSIFICATION
  if $Y_R < Q_L$ **then**
    Y_Q = **Low Effort**
  else if $Y_R \geq Q_H$ **then**
    Y_Q = **High Effort**
  else Y_Q = **Moderate Effort**
  end if
  return (Y_R,Y_Q)
end procedure

vector. After selecting the features from each dataset, Algorithm 2 is applied to predict development effort and its class as high, moderate and low.

IV. Experiments and Results

The data sets considered for validating our proposed approach and data preprocessing are discussed in Section IV-A. The different performance measures used to compare prediction performance are presented in Section IV-B. The experimental results are discussed in Section IV-C.

A. Data sets and Data Preprocessing

We have collected six data sets namely as Albrecht, China, COCOMO81, Desharnais , Kemerer and Maxwell from GITHUB and the PROMISE Repository [16] as stated in [1]. We have applied Min-Max normalization on all the datasets collected.

Min-Max Normalization: It is a commonly used data normalization technique that converts a large range of numbers into a small range of numbers.

$$Normalization(smt) = \frac{smt_i - min(smt)}{max(smt) - min(smt)} \quad (5)$$

where smt_i is the *ith* value of the software metric *smt* and *min(smt)* and *max(smt)* represent the minimum and maximum values of the software metric *smt* respectively.

B. Performance Measures

We have used Mean Absolute Error (MAE), Balanced Magnitude of Relative Error (BMMRE) and Adjusted R^2 [1],as the performance measures to validate our proposed approach for Output 1,i.e, software development effort. For Output 2, we have used performance measures such as Accuracy, Precision, Recall and F1-score.

The performance parameters for Output 1 are defined as follows:

- **MAE:** MAE is the mean of the absolute values of the individual prediction errors of all the samples which are present in the testing data. Prediction error is nothing but the difference between the Actual Effort(E_{act}) and the Predicted Effort(E_{pred}) for a particular sample.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|E_{act_i} - E_{pred_i}\right| \quad (6)$$

  where n is the number of samples in the dataset.
- **BMMRE:**

$$BMMRE = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\left|E_{act_i} - E_{pred_i}\right|}{\min\left(E_{act_i}, E_{pred_i}\right)}\right) \quad (7)$$

  where, N is the number of samples in the dataset and $E_{pred\ i}$ and $E_{act\ i}$ are the predicted and actual effort values for the i^{th} sample in the dataset respectively.
- **Adjusted R^2:**
  Adjusted R^2 shows if the addition of additional predictor variables improves the performance of a regression model or not. Essentially, closer the value of Adjusted R^2 to 1,better the model.

$$Adjusted\ R^2 = 1 - \frac{(1-R^2)(n-1)}{n-p-1} \quad (8)$$

  where, number of samples in the dataset is n, number of predictor variables is p and multiple coefficient of determination is R^2.

To classify our predicted effort into 3 broad classes(Low,Moderate and High), the confusion matrix for the same would have a 3x3 dimension. Following are the parameters we considered:

- **Accuracy**: It gives us a measure of the number of samples correctly classified by the classifier.
- **Precision:** It gives us a measure of what fraction of predicted positive examples were actually positive.
- **Recall:** It gives us a measure of what fraction of all positive examples were actually positive.
- **F1-Score:** This measure essentially combines precision and recall into a single metric to give equal importance to each class.

C. Discussion of Results

In this section, we have compared the results of our proposed approach with the results reported in [1]. In Table II, we have compared the performance of LA with other feature

selection techniques such as GSA,BF, SSFS, LFS and RS as used in [1] based on Mean Absolute Error(MAE). From the LAFS column of Table II, it is clear that LA based feature selection outperforms the other techniques and hence yields much lower magnitudes of Mean Absolute Error(MAE) as compared to the other techniques, with a maximum error of 17.20 percent for the Kemerer dataset and minimum error of 4.06 percent for the China dataset. Hence, the values of the table clearly indicate that LA based SEE is more effective than other techniques.

In Tables III and IV, we have compared the performance of the proposed model with other techniques on the basis of BMMRE and Adjusted R^2 receptively. In Tables II,III and IV, the results are reported using the **Linear Regression model** because compared to the other regression models,it yields a better performance considering the performance metrics specified in the previous section. In Table III, we found that LAFS yields much lower magnitudes of BMMRE as compared to most techniques of feature selection. However, GSA based SEE slightly outperforms our proposed model in the COCOMO81 dataset. Lower values of BMMRE indicate lesser magnitude of error and better model performance. om the values in the LAFS column of Table IV, we found that LA based SEE has the best performance among the other feature selection methods. The maximum Adjusted R^2 value reported is 0.95 for the China dataset and the minimum Adjusted R^2 value reported is 0.59 for the Kemerer dataset using LA based SEE.

In Table V, we have presented the optimal number of features selected by LA based feature selection for each of the datasets - Albrecht, China, COCOMO81, Desharnais , Kemerer and Maxwell. We observed that the LA based feature selection technique is successful in removing many redundant features in the Albrecht, China, Desharnais, Cocomo81 and the Kemerer datasets with the highest number of features being removed in the China dataset (9 features). In Tables VI,VII and VIII we have presented the performance of our model using the other regression models which include : LASSO regression, ElasticNet regression and Ridge regression on the basis of MAE, BMMRE and Adjusted R^2.

Table IX presents the scores obtained by plotting the confusion matrices for each of the datasets using the regression model which performs the best for that particular dataset. Analyzing Table IX , we found that, out of the 6 datasets considered the Linear Regression model performs the best in the Albrecht and Desharnais datasets with classification accuracies as 75% and 63% respectively. However, the

TABLE II
COMPARISON OF LA-BASED FEATURE SELECTION BASED ON MAE(%)

Dataset	BF [1]	SSFS [1]	GSA [1]	LFS [1]	RS [1]	LAFS
Albrecht	21.1	14.5	14.1	21.2	34.2	10.06
China	20.2	20.2	11.6	20.2	35.3	4.06
COCOMO81	27.1	27.1	15.7	27.1	16.9	8.06
Desharnais	28.8	29.8	12.4	29.8	33.2	9.26
Kemerer	33.6	41.1	33.6	33.6	35.6	17.20
Maxwell	33.7	45.9	17.5	33.7	46.8	11.50
MEAN	27.4	29.7	17.4	27.6	33.6	10

TABLE III
COMPARISON OF LA-BASED FEATURE SELECTION BASED ON BMMRE

Dataset	BF [1]	SSFS [1]	GSA [1]	LFS [1]	RS [1]	LAFS
Albrecht	35.5	27.2	13.2	24.7	34.9	1.29
China	20.9	25	10.6	31.9	34.1	1.65
COCOMO81	34.2	27.9	14.6	15.9	18.9	15.30
Desharnais	18.6	19.6	14.5	23.4	19.2	10.11
Kemerer	17.9	20.8	19.1	20.7	18.8	1.36
Maxwell	39.6	46.6	27.5	49.9	41.2	10.23
MEAN	27.7	27.8	16.6	27.6	27.9	6.7

TABLE IV
COMPARISON OF LA-BASED FEATURE SELECTION BASED ON ADJUSTED R^2

Dataset	BF [1]	SSFS [1]	GSA [1]	LFS [1]	RS [1]	LAFS
Albrecht	0.44	0.57	0.71	0.52	0.48	0.84
China	0.60	0.28	0.78	0.45	0.17	0.95
COCOMO81	0.71	0.58	0.70	0.56	0.37	0.89
Desharnais	0.64	0.64	0.77	0.59	0.60	0.89
Kemerer	0.58	0.55	0.62	0.49	0.48	0.59
Maxwell	0.57	0.57	0.75	0.75	0.61	0.79
MEAN	0.59	0.53	0.72	0.56	0.45	0.82

TABLE V
OPTIMAL NUMBERS OF FEATURES SELECTED BY LAFS

Dataset	Total Input Features	Optimal Features
Albrecht	7	4
China	16	8
COCOMO81	16	14
Desharnais	10	7
Kemerer	6	4
Maxwell	25	24

TABLE VI
COMPARISON OF ALL REGRESSION MODELS (MAE(%)) USING LAFS

Dataset	Linear	LASSO	ElasticNet	Ridge
Albrecht	10.06	9.71	9.05	9.21
China	4.06	6.77	6.34	3.86
COCOMO81	8.06	6.97	6.01	7.99
Desharnais	9.26	9.95	8.99	9.26
Kemerer	17.20	12.79	12.89	16.06
Maxwell	11.50	7.44	7.28	11.28

TABLE VII
COMPARISON OF ALL REGRESSION MODELS (BMMRE) USING LAFS

Dataset	Linear	LASSO	ElasticNet	Ridge
Albrecht	1.29	1.84	1.35	2.04
China	1.65	5.86	5.23	1.87
COCOMO81	15.30	10.31	18.48	15.23
Desharnais	10.11	15.01	8.62	7.36
Kemerer	1.36	1.09	0.99	1.22
Maxwell	11.50	23.74	20.38	10.63

TABLE VIII
COMPARISON OF ALL REGRESSION MODELS (ADJUSTED R^2) USING LAFS

Dataset	Linear	LASSO	ElasticNet	Ridge
Albrecht	0.84	0.85	0.86	0.85
China	0.95	0.92	0.93	0.95
COCOMO81	0.89	0.90	0.91	0.89
Desharnais	0.89	0.88	0.89	0.89
Kemerer	0.59	0.70	0.92	0.62
Maxwell	0.79	0.87	0.87	0.80

TABLE IX
PERFORMANCE MEASURES OF REGRESSION MODELS FOR OUTPUT 2 (EFFORT CLASS)

Dataset	Model	Class	Accuracy	Precision	Recall	F1-Score
Albrecht	Linear	L	83.33	0.67	0.67	0.67
		M	91.67	0.83	0.83	0.83
		H	75	0.75	0.75	0.75
			Overall: 75%			
China	Ridge	L	86.77	0.74	0.74	0.74
		M	87.98	0.76	0.76	0.76
		H	74.75	0.75	0.75	0.75
			Overall: 74.7%			
COCOMO81	ElasticNet	L	77.78	0.56	0.56	0.56
		M	80.95	0.63	0.63	0.63
		H	74.60	0.74	0.74	0.74
			Overall: 66.7%			
Desharnais	Linear	L	82.72	0.67	0.67	0.67
		M	77.78	0.57	0.57	0.57
		H	65.43	0.64	0.64	0.64
			Overall: 63%			
Kemerer	ElasticNet	L	86.67	0.75	0.75	0.75
		M	100	1.0	1.0	1.0
		H	86.67	0.86	0.86	0.86
			Overall: 86.7%			
Maxwell	ElasticNet	L	87.10	0.75	0.75	0.75
		M	80.65	0.63	0.63	0.63
		H	67.74	0.67	0.67	0.67
			Overall: 67.7%			

ElasticNet regression model slightly outperforms the linear regression model with respect to classification accuracies in the COCOMO81 (66.7%), Maxwell (67.7%) and Kemerer (86.7 %) datasets. The Ridge regression model yields the best classification accuracy for the China dataset (74.7%).

V. CONCLUSION

In this paper, we have proposed a learning automata based regression model to predict software development effort in duplex mode. We have predicted both the software development effort and the effort class as high, moderate and low. We have used learning automata to select the suitable features for the model.We have applied our proposed approach on six data sets and have compared our results with some existing models. We observed that learning automata is able to select the features properly and using the selected features, the regression models are able to provide better prediction accuracy than other existing models. More robust feature selection technique may be used to select the relevant features for software development effort prediction in duplex form. In future, the concept of learning automata may appled on non-linear regression models such as a deep learning network to predict software development effort in duplex mode.

REFERENCES

[1] Solomon Mensah, Jacky Keung, Michael Franklin Bosu, Kwabena Ebo Bennin, Duplex Output Software Effort Estimation Model with Self-Guided Interpretation, Information and Software Technology, Volume 94, 2018,ISSN 0950-5849, Pages 1-13.

[2] Yu Su, Kaiyue Qi, Chong Di, Yinghua Ma, Shenghong Li, Learning Automata based Feature Selection for Network Traffic Intrusion Detection, IEEE Third International Conference on Data Science in Cyberspace, 2018.

[3] Zhang Y, Di C, Han Z, An Adaptive Honeypot Deployment Algorithm Based on Learning Automata, IEEE Second International Conference on Data Science in Cyberspace, IEEE Computer Society, 2017, Pages 521-527.

[4] Barry Bohem, Chris Abts, A. Winsor Brown, Sunita Chulani, Bradford K. Clark, Ellis Horowitz, Ray Madachy, Donald J. Reifer, Bert Steece, Software Cost Estimation with COCOMO II(with CD-ROM), Englewood Cliffs, NJ:Prentice-Hall, 2000, ISBN 0-13-026692-2

[5] Whigham, P.A., Owen, Caitlin, MacDonell, Stephen. (2015), A Baseline Model for Software Effort Estimation, ACM Transactions on Software Engineering and Methodology 24(3), Pages 1-11.

[6] R. Tibshirani, Regression Shrinkage and Selection via the LASSO: A Retrospective, J. R.Stat. Soc.: Ser. B (Stat. Methodol.) 73(3)(2011), Pages 273–282.

[7] H. Zou, T. Hastie, Regularization and Variable Selection via the Elastic Net, J. R. Stat.Soc.: Ser. B (Stat. Methodol.) 67 (2) (2005), Pages 301–320.

[8] A.E. Hoerl, R.W. Kennard, Ridge Regression: Biased Estimation for Nonorthogonal Problems, Technometrics 12 (1) (1970) Pages 55–67.

[9] Y. Miyazaki, M. Terakado, K. Ozaki, H. Nozaki, Robust Regression for Developing Software Effort Estimation Models, J. Syst. Softw. 27 (1) (1994), Pages 3–16.

[10] Menzies T., Yang Y., Mathew G. ,Negative Results for Software Effort Estimation, Empirical Software Eng 22(2017), Pages 2658–2683.

[11] Mensah S., Keung J., Bennin K.E., Bosu M.F. , Multi-Objective Optimization for Software Testing Effort Estimation, In Proceedings of the International Conference on Software Engineering and Knowledge Engineering (SEKE) , Vol. 2016-January, Pages 527-530.

[12] Keung, Jacky, Kocaguneli, Ekrem, Menzies, Tim, Finding Conclusion Stability for Selecting the Best Effort Predictor in Software Effort Estimation, Automated Software Engineering(2013), 20, Pages 543–567.

[13] C. Lokan, E. Mendes, Investigating the Use of Moving Windows to Improve Software Effort Prediction: A Replicated Study, Empirical Softw.Eng., 22 (2) (2016), Pages 716–767.

[14] K. Dejaeger, W. Verbeke, D. Martens, B. Baesens, Data Mining Techniques for Software Effort Estimation: A Comparative Study, IEEE Trans. Softw. Eng., 38 (2) (2012), Page 375–397.

[15] M. A. Shah, D. N. A. Jawawi, M. A. Isa, M. Younas, A. Abdelmaboud and F. Sholichin, "Ensembling Artificial Bee Colony With Analogy-Based Estimation to Improve Software Development Effort Prediction," in IEEE Access, vol. 8, pp. 58402-58415, 2020, doi: 10.1109/ACCESS.2020.2980236.

[16] T. Menzies, R. Krishna, D. Pryor, The Promise Repository of Empirical Software Engineering Data, North Carolina State University, Department of Computer Science, 2016.