Received: December 19, 2018

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Effort Estimation in Agile Software Development using Evolutionary Cost-Sensitive Deep Belief Network

Hosahalli Mahalingappa Premalatha[1]* **Chimanahalli Venkateshavittalachar Srikrishna[2]**

[1]*People's Education Society University, India*
[2]*People's Education Society Institute of Technology, India*
* Corresponding author's Email: premalathaphd2017@gmail.com

Abstract: In today's scenario, frequent requirement changes in software development are a notable issue in the software field. Because of the frequent changes, fulfilling the user's requirement is very difficult. As a solution to such issues, Agile Software Development (ASD) has efficiently replaced the traditional methods of software development in industries. Due to various aspects of ASD, it is extremely hard to follow, maintain and estimate the general item. Hence, in order to tackle the Effort Estimation Problem (EEP) in ASD, various types of EEP have been identified in existing methods. The Evolutionary Cost-Sensitive Deep Belief Network (ECS-DBN) model implemented in this paper for effort prediction in any agile technique. The ECS-DBN method has no impact on agility because it uses simple and small inputs. The proposed method used in planning stage of software development to support the project managers in further development of agile software. The project managers characterize the structure of the ECS-DBN, while the parameter estimation consequently gained from a dataset. This paper used different statistics like accuracy, prediction at m level to evaluate the accuracy of the model. The ECS-DBN method achieved nearly 99% accuracy compared to the existing methods.

Keywords: Agile projects, Effort estimation, Evolutionary cost-sensitive deep belief network, Software development, Planning stage.

1. Introduction

Nowadays, software is significantly used in many applications like home appliances, nuclear-power-plants, automobiles, telecommunications, medical devices and so on [1]. The software testing process is an important task in developing software to make it free from bugs and defects and additionally, it improves the quality of software. The software quality estimation uses several factors such as reliability, efficiency, software functionality, testability and so on. In these quality factors, software reliability is a more significant factor because it checks how far software is consistent by tolerating failures during the lifetime of software [2, 3]. The definition of software reliability is the successful running of the system with no error at a particular time period. To make the system more efficient with less error and less maintenance, there is a need of predicting and estimating software reliability using recent techniques and methodologies. Earlier researches focused on to reduce complexities and the failure rate in the system. It is a very challenging process for existing methods to calculate the finite cost in a large area with a population at the random movement of many components. [4]. At present, the software testing takes more time and cost and it makes the SD process an expensive task. But, the cost of testing decreases with the reduction of testing time [5]. However, most of the software delivered without enough testing, which is due to marketing pressures and the aim to save testing time and cost, but delivering a software without sufficient testing may lead to loss of revenue [6, 7]. The software testing is an essential technique to develop a bug-free software and it is very helpful for software developers. Several existing research works implemented to improve the quality of the software.

The process of determining the best predicted effort required to develop a software project is termed as software EE. This estimation can be divided into three levels: First stage consists of size estimation, the second stage involves EE and the third stage is cost estimation, the EE is calculated in terms of PM (Person-Months) [8-9]. ASD process is an iterative and incremental approach, which improves the overall product development routine by using customer feedback and continuous evaluation of documentation projects. Agile methodology plays a bigger role in implementing communication goals between customers in a planned manner. The main issue of determining EE in agile methods is to focus on the effort and degree of difficulties of teamwork rather than an individual [10]. Some of the existing work showed that while using ASD, development team used story point approach [11] to calculate the effort with the help of user story and project velocity as inputs. The main task is to estimate the required effort to reduce the technical complexity of the project. Agile software has many advantages, as it is iterative, modular, incremental, customer oriented, and time-bound [12]. Therefore, the main objective of this research is to implement a method that will facilitate the assessment of the required effort. The proposed ECS-DBN method helped overcome the problems of traditional effort, cost prediction models because the ECS-DBN method can able to run the model with missing data, also reflect causal relationships. The ECS-DBN is very flexible compared to other existing methods, because the method purely based on empirical data, expert judgement or the combination of both. This proposed method is suitable in the planning stage, even before the development of software, and it helps project managers in further ASD. It should not affect the agility and it should be suitable for any agile method. The prediction accuracy is determined by various statistics in software estimation. The most commonly used metrics are the Magnitude of Relative Error (MRE), the Mean Magnitude of Relative Error (MMRE), and the Prediction at Level m (Pred. (m)).

This paper is composed as follows: the investigation of existing models for software Effort Prediction (EP) is described in Section 2, ECS-DBN is explained in Section 3, the experimental results are given in Section 4. The conclusions as well as the outlines of future work are presented in Section 5.

2. Literature review

Considerable effort is missing in the domain of EE for ASD and most of the researchers have used traditional EE techniques for determining the software effort that provides inaccurate results. This section presented a brief valuation of some essential contributions to the existing literatures.

O. Malgonde, and K. Chari, [13] explored a critical aspect of agile development, i.e., EP, that cuts across these tools and agile project teams. The work developed a model for story-EP uses variables that were readily available when a story was created. The method used seven predictive algorithms to predict a story's effort and developed an Ensemble-based Learning Method (EL) for predicting story effort. The experimental result demonstrated the approach by optimizing sprint planning for two projects from our dataset using an optimization model. But the experiments were limited to the dataset which has the potential of impacting the generalizability of findings. This is an inherent limitation; moreover, the method provided limited conclusions on the superior performance of the prediction methods and restrict the inferences to EE in ASD projects.

P. Xiao, B. Liu, and S. Wang [14] developed an improved feedback-based defect prediction strategy, which combined the defect prediction with the feedback control mechanism during the Software Testing Process (STP) to address the problem of ranking optimization. In addition, a novel approach called feedback-based integrated prediction (FIP) was proposed to improve the prediction accuracy, where a global predictor and a local predictor were employed to make an integrated prediction using the weight to adjust the effects of predictors at different test stages. The performance of FIP was investigated on 10 public datasets by systematic experiments, which stated that FIP had better robustness and prediction efficiency when compared to traditional DP. The experimental results showed that FIP had a strong tolerance for defect misclassification, although false positive can affect the prediction performance of FIP

V. Nguyen, B. Boehm, and L. Huang [15] investigated the use of moving windows to determine relevant training data for COCOMO calibration. This method presented a windowing calibration approach and accessing the performance of EE models to calibrate the COCOMO, windows and all data. This study provided empirical evidence to support the use of small windows of completed projects to calibrate models when COCOMO-like data was available. Additionally, when the change in SD over time was rapid, the use of windows was more justifiable for improving estimation accuracy. If the windows large in size, then this method produced worse estimations.

Z. W. Zhang, X. Y. Jing, and T. J. Wang [16] implemented a novel Non-Negative Sparse Graph based Label Propagation approach (NSGLP) for

semi-supervised learning in software defect prediction. The NSGLP improved the generalization capability by using few labeled data and abundant unlabeled data. The class-balance labeled training dataset was constructed by NSGLP method and learn a sparse graph for characteristics of defect data by Laplacian score sampling and sparse representation. The NSGLP method used the constructed graph to predict the labels of the unlabeled software modules through label propagation approach. The NSGLP method provided better performance compared to existing semi-supervised software prediction task on ten NASA datasets. But, the proposed method leads poor performance in defect prediction due to insufficient labeled data.

M. Boopathi, R. Sujatha, C. S. Kumar, and S. Narasimman [17] proposed a hybrid technique, namely Markov chain and Artificial Bee Colony (ABC) optimization methods were used to achieve the software code coverage. A number of paths were generated using Linear-Code-Sequence-And-Jump (LCSAJ) coverage. The LCSAJ was employed to decrease the number of independent paths as related to the paths generated by original path testing. The qualities of test cases enhanced in each iteration of ABC optimization and determined the sequences of complete LCSAJ independent paths in a software code. The calculation of test tolerability and reliability of different kind of critical software is difficult to through ABC optimization with mutation testing.

S. Bilgaiyan, S. Mishra, and M. Das [18] focused on as hybrid Neural Network (NN) namely, Feed Forward Back Propagation (FFBP) and Elman NN (ENN) to solve the EEP. These hybrid method was applied to a dataset that consists of 21 projects based on ASD from 6 different software houses. The limitation of the proposed method was not performed well for other datasets from heterogeneous software development methods.

P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski [19] developed an ensemble three machine learning such as Support Vector Machine (SVM), NN and Generalized Linear Models (GLM). These models were intended to serve as a decision support tool for any organization developing and implementing software systems regardless of the industry sector where incorrect estimation may lead to negative implications. But, the limitation was that the method supported only small group industries, whereas the method provided poor performance in large scale industries for EEP.

The proposed ECS-DBN method is implemented to overcome the above issues addressed by the existing methods, and to estimate the effort costs automatically to improve the performance of cost-sensitive DBN.

3. Proposed method

Agile methods avoid the formalisms of traditional specification and design techniques. The downside of this is a lack of specification metrics for project planning. At the same time, agile project managers have to plan their projects as any other traditional project manager for the EP. The main purpose of this paper is to build a ECS-DBN which can help agile project managers to predict project effort. The basic steps of the proposed method are depicted in Fig. 1.

3.1 Cost-sensitive deep belief network

The main aim of cost-sensitive learning is to reduce the overall cost of the training dataset. The cost of misclassifying x as class j when x actually belongs to class i, denotes the sample data as $x, C_{i,j} \in [0,1]$ for total number of classes K. In addition, $C_{i,j} = 0$, when $i = j$, which indicates the cost of correct classification is 0.

Given the misclassification costs $C_{i,j}$, a data sample should be classified into the class that has the minimum expected cost. Based on decision theory, the decision rule minimizes the expectation cost of $R(i|x)$ for classifying an input vector x into class i can be expressed as in Eq. (1):

$$R(i|x) = \sum_{j=1,j\neq 1}^{K} P(j|x) C_{i,j} \quad (1)$$

Where $P(j|x)$ is the posterior probability estimation of classifying a data sample into class j. Given the prior probability $P(x_n)$, the general decision rule indicates which action to take for each data sample x_n, thus the overall risk R is described in Eq. (2).

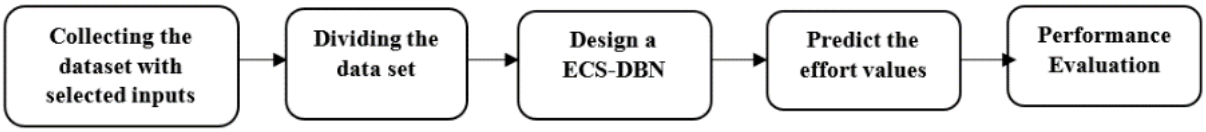

Figure. 1 Structure of the proposed ECS DBN method

$$R = \sum_{n=1}^{N} \sum_{i=1}^{K} R(i|x_n) P(x_n) \quad (2)$$

Based on Bayes hypothesis, a perfect classifier will give a choice by computing the desire risk of grouping a contribution to each class and predicts the mark that achieves the minimum overall expectation risk. The classification error penalties are described by misclassification costs. In cost-sensitive adapting, all misclassification costs are basically non-negative. Mathematically, the probability that a sample data $x \in S$ belongs to a class$j$, a value of a stochastic variable y, can be expressed as in Eq. (3):

$$P(y = j|x) = softmax_j(b + Wx) \quad (3)$$

The misclassification edge esteems acquainted with transforming posterior probabilities into class labels to such that the misclassification costs are limited. Introducing the misclassification limit esteem $1 - C_{i,j}$ on the obtained posterior probability $P\ (y = j|x)$, one can obtain the new probability P_{ξ} that are described in Eq. (4):

$$P_{\xi}(y = j|x) = P(y = j|x).\left(1 - C_{i,j}\right) \quad (4)$$

Generally, the misclassification threshold values for minority classes are larger than majority classes. The hypothesized prediction $f(x)$ of the sample x is the member of the maximum probability among classes, can be obtained by using the following Eq. (5):

$$F(x) = \underset{j}{argmax}\, P_{\xi}(y = j|x) \quad (5)$$

The proposed cost-delicate learning technique only concerns about the yield layer of a DBN. For imbalanced order issues, the earlier probability dispersion of various classes is basically imbalanced or non-uniform. To reflect class imbalance, there is a need to present the misclassification cost at the yield layer to mirror the imbalanced class dispersions. Moreover, conventional preparing calculations generally assume uniform class dispersion with equivalent misclassification costs, $i.e. \forall i, j \in [1, 2, \cdots, K], if\ i = j;\ C_{i,j} = 0, if\ i \neq j;\ C_{i,j} = 1$, which isn't valid in some real-time applications. To maintain a strategic distance from hand tuning of misclassification costs, adaptive DE calculation is executed in this paper.

Adaptive DE calculation is a basic successful and proficient developmental calculation which could acquire ideal arrangement by advancing and updating a population of individuals during several generations. It endeavors to adaptively self-update the control parameters without the need of earlier learning.

3.2 Evolutionary cost-sensitive deep belief network

Evolutionary Algorithm (EA) is broadly utilized for optimization calculation which is driven by the natural advancement process. The EA calculation can be intended to streamline the misclassification costs that are unknown. In this paper, ECS-DBN is proposed by combining cost-sensitive capacity into its characterization with the misclassification costs through adaptive differential advancement. The initial step is to select the cost of misclassification randomly, then train a DBN with the training dataset. According to execution on preparing dataset, appropriate misclassification costs are chosen to generate the population of the next generation. In this generation, crossover and mutation administrators are utilized to enhance the new population for misclassification costs. To reach the maximum number of generations, Adaptive DE calculation continuously iterates the next generation between selection and mutation.

3.2.1. Chromosome encoding

Chromosome encoding is a major process in EA that aims for effective representations for important variables for better execution. In some applications, misclassification costs in DBN are generally obscure. Therefore, in proposed approach every chromosome describes the misclassification costs for various classes to obtain the appropriate expenses. The misclassification costs for ECS-DBN choose the best chromosome from the last developed method. The chromosome encoding here straightforwardly encodes the misclassification costs as qualities in the chromosome with numerical type and esteem scope of [0, 1].

3.2.2. Population initialization

The initial population received by means of consistently random testing in feasible solution space for every factor inside the predetermined scope of the relating variable. The population hold conceivable misclassification expenses and structures the unit of development. The development of the misclassification costs is an iterative process with the population in each iteration cycle is called generation.

3.2.3. Adaptive DE operators

After initialization, adaptive DE evolves the population with a sequence of three evolutionary operations, i.e. mutation, crossover, and selection, generation by generation. Mutation is carried out with DE mutation strategy to create mutation individuals based on the current parent population. After mutation, a binomial crossover operation is utilized to generate the final offspring. In adaptive DE, each individual has its associated crossover probability instead of a fixed value. The selection operation selects the best one from the parent individuals and offspring individuals according to their corresponding fitness values. In this way, the control parameters are automatically updated with the appropriate values without the need of prior parameter setting knowledge in DE.

3.2.4. Fitness evaluation

Fitness evaluation enables us to pick the suitable misclassification costs. In this proposed strategy, every individual chromosome is introduced into individual DBN as misclassification costs. At this point, it creates suitable misclassification costs for DBN by utilizing the preparation set. G-mean of preparing set picked as the target work for the process of optimization.

3.2.5. Termination condition

EA are designed to evolve the population generation by generation and maintain the convergence as well as diversity characteristics within the population. A maximum number of generations set to be a termination condition of the algorithm. In this implementation, the solutions converged when the best fitness value remains unchanged over the past 30 generations. The algorithm stops once it reaches the maximum number of generations or meets the convergence condition.

3.3 ECS-DBN creation

The misclassification costs are used to form an ECS-DBN, when the process of optimization ends with best individual. The best individual is obtained from the last generation. Initially, the intention was to build a ECS-DBN model for sprint (iteration) EP. The model was planned to use the existing database of software projects for model validation. These are the projects of a micro software company, which used agile methods for several years. The scrum master has two datasets,

- The list of software entities with complexity classification extracted from the project log for each task [20].
- The list of knowledge and skills calculated for each developer, including motivation and experience, classified into 5 levels (from 1, very low level, to 5, very high level), and updated twice a year [21].

When planning new tasks, the project developers and the project manager try to find the best solution for both sides: developers' ideas and project productivity. When working on a task, the developers record their working hours and the types of activities (that takes only 1-2 minutes at the end of a workday). So, they always know the real task and project status. Consequently, the authors realized very early in the process that it was convenient to build an ECS-DBN model for task EP because it is easier to collect input data which are less complex. It is easier for a project manager to estimate a node value (e.g. whether the complexity of a report is "low", "medium" or "high") and easy to predict the iteration duration based on each developer's prediction effort.

4. Experimental result

The proposed ECS-DBN method has been implemented in Java NetBeans 8.2 version, 32 bit operating system and 8GB RAM. The data used in this research originated from agile projects of a small software company. In order to evaluate the efficiency of the proposed system an evaluation metric is employed. The MMRE and the Pred. (m) are the two important metrics that are used in this research to assess the accuracy of prediction in software estimation. But, in this research, there is only one prediction per one task used for predicting the performance of ECS-DBN.

Therefore, other statistical measures are also used to assess the accuracy of this model: Relative Absolute Error (RAE), Accuracy, Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Root Relative Squared Error (RRSE). These measures are chosen due to their simplicity of use and because of their application areas [22, 23].

4.1 Dataset description

The data collected from the small software company for agile projects to estimate the effort task. But, these data are not suitable for direct use in the ECS-DBN model. That must be prepared, but it requires more time to process the training data, so the data are separated in three datasets. The first dataset consists of 40 tasks, the second of 50 tasks, and the

third of 70 tasks. Tasks are grouped chronologically based on creation time. Grouping made neither by size, nor by complexity, nor according to the developer who performs the task, but it completely based on the creation time. All the datasets include the tasks of different duration and complexity, created by different developers.

4.2 Performance evaluation

Numerical measures are the most commonly used measures to evaluate and validate fault prediction models. The detail of these measures is given below.

4.2.1. Mean magnitude of relative error (MMRE)

MMRE is the average of the MREs calculated over all the reference tasks, which are given in the Eq. (6).

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i \quad (6)$$

MRE represents the normalized measure of deviations between the actual and the estimated values that are depicted in Eq. (7):

$$MRE = \frac{|y_i - f(x_i)|}{y_i} \quad (7)$$

Where, x_i is the predict value/actual value and y_i is an observed value.

4.2.2. Prediction at level m

MMRE is the measure of standard deviation (spread) of a variable x_i, while Pred. (m) is the measure of peak (kurtosis) of a variable x. MMRE prefers models that predict estimates below the mean, and it is also sensitive to outliers. A sensitivity to outliers allows MMRE to detect the model occasionally tends to be very inaccurate. Pred. (m) measures the percentage of estimates that are within m percent of the actual values. It is usually set to m = 25. Pred. @(25%) detects what percentage of estimates is within a tolerance of 25%.

4.2.3. Accuracy

Accuracy is the percentage of the correctly classified samples over the total number of samples. The accuracy can range from 0% to 100%. The general formula for accuracy can be given in Eq. (8),

$$accuracy = \frac{TN+TP}{TN+TP+FN+FP} \quad (8)$$

Where, TN is True Negative, TP is True Positive, FN is False Negative and FP is False Negative.

4.2.4. Mean absolute error

MAE is the average of the absolute values of the prediction errors, given by the Eq. (9):

$$MAE = \frac{1}{n}\sum_{n}^{1}|f(x_i - y_i)| \quad (9)$$

Where n is the number of predictions, $f(x_i)$ is a predicted value. All the errors are weighted equally due to linear score.

4.2.5. Root mean square error

RMSE is another measure of deviation between the predicted $f(x_i)$ and the real valuey_i. RMSE is the square root of MSE that predict the mean relative error. The formula for RMSE is given in Eq. (10)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2} \quad (10)$$

Large errors are weighted more heavily because the errors are averaged after they squared. The error variance can be detected if RMSE and MAE are used together. The variation is greater if the difference between them is larger. If all the errors have the same magnitude, MAE = RMSE, otherwise, RMSE > MAE.

Both MAE and RMSE are useful for the comparison of the prediction errors of different models for a particular variable and not for the comparison between variables. That shows errors in the same unit and scale as the parameter itself, so they are scale-dependent.

The method also includes measures that can be used for the comparison of models those errors are measured in different units. Such measures include in Eqs. (11) and (12):

$$RAE = \frac{\sum_{i=1}^{n}(|f(x_i - y_i)|)}{\sum_{i=1}^{n}(\bar{y}_i - y_i)} \quad (11)$$

$$RRSE = \sqrt{\frac{\sum_{i=1}^{n}\left(f(x_i - y_i)\right)^2}{\sum_{i=1}^{n}(\bar{y}_i - y_i)^2}} \quad (12)$$

Where $\bar{y}_i$ is the mean value of$y$.

4.3 Parameter evaluation

The experiments were conducted to evaluate the performance parameters such as MMRE, Pred. m, accuracy, MAE, RMSE, RAE and RRSE. The prediction accuracy for all the datasets is shown in Table 1 that represents the performance of parameters of proposed ECS-DBN.

Table 1. Results of various parameters of proposed method

Number of tasks	40 (first)	50 (second)	70 (last)	90 (40+50)	160
Accuracy	92%	97%	97.86%	97%	99.487%
MAE	0.1056	0.0521	0.0520	0.0454	0.023
RMSE	0.190	0.1298	0.1121	0.111	0.054
MSE	0.0435	0.0360	0.033	0.033	0.0232
RAE	35.18%	24.20%	18.09%	17.25%	9.23%
RRSE	50.27%	40%	30.95%	32.31%	17.21%
Pred. (25)%	100%	100%	100%	100%	100%
Pred. (10)%	93%	100%	100%	100%	100%
MMRE	12.50	2.94	4.21	6.59	6.10

For a dataset of 160 instances, only the effort of one task is wrongly classified. The true effort for the last set of data will be 2.4% which is indicated by the values of MAE. The error variance is relatively small that is indicated by the small differences between RMSE and MAE values. The Pred. (m) (i.e. 10 and 25%) and the MMRE metrics are also applied to this dataset. Since the ECS-DBN model estimates effort as a set of probability distributions of all possible classes, a Conversion method is used to obtain the estimated effort as a discrete value. The class probabilities should be normalized, so that their sum equals one. The estimated effort is then given by the Eq. (13):

$$Effort = \sum_{i=1}^{n} \rho_{classi} \mu_{classi} \quad (13)$$

Where μ_{classi} is the mean of class i, and ρ_{classi} is its respective class probability.

4.4 Comparative analysis

The proposed method ECS-DBN are compared with existing methods such as hybrid FFBP and ENN [18] and ensemble machine learning such as SVM, and GLM [19]. S. Bilgaiyan, *et al.*, [18] focused on two types of ANN- FFBP and ENN to solve the EEP. The FFBP-ENN method has high computation speed, fixed computation time and fault tolerance with respect to Elman network. The FFBN-ENN method didn't perform well in other datasets collected from heterogeneous SD methods, which was considered as a limitation of this method. The Table 2 describes the comparison results of proposed with existing methods in terms of MMRE, Pred. (25) and RMSE.

P. Pospieszny, *et al.*, [19] aimed to narrow the gap between up-to-date research results and implementations within organizations by proposing effective and practical machine learning deployment. This was achieved by smart data preparation and applying ensemble averaging of three machine learning algorithms (SVM, NN and GLM) on ISBSG dataset. The limitation was the impact of software sizing especially on EE. As an input variable, it has the most significant impact on forecasting the mentioned output parameter. The hybrid method [ENN+FFBP] achieved nearly 14.90 MMRE and 13.49 MMRE, whereas the proposed ECS-DBN achieved 12.50 MMRE by using evolutionary algorithm. The performance measure like Pred. 25% and MSE for existing methods achieved 95.23% and 0.052 in FFBN, whereas 61.96% and 0.13 error rate achieved by GLM method. But the ECS-DBN achieved 100% in Pred. 25% and achieved very low error rate as 0.043 in MSE. Compared to the existing hybrid method [SVM+GLM], the ECS-DBN method achieved very low RMSE value. The GLM achieved more RMSE value nearly 0.35, but the proposed method achieved 0.19 RMSE value with low error rate. The overall experimental results stated that the proposed method achieved better performance than existing methods such as ENN+ FFBP and SVM+GLM in terms of MMRE, Pred. 25% and RMSE.

Table 2. Comparison analysis of ECS-DBN with existing methods

Authors	Techniques	MMRE	Pred. (25%)	RMSE	MSE
O. Malgonde, *et al*, [13]	EL	-	-	15.3	3.911
S. Bilgaiyan, et al., [18]	ENN	14.80	94.86	-	0.056
	FFBP	13.49	95.23	-	0.052
P. Pospieszny, et al., [19]	SVM	13	76.91	0.27	0.07
	GLM	18	61.96	0.35	0.13
Proposed	ECS-DBN	12.50	100	0.19	0.0435

5. Conclusion

This paper developed an ECS-DBN model for EP in ASD projects. The ECS-DBN model is relatively small and simple and all the input data are easily elicited, so that the impact on agility is minimal. The model predicts task effort, and it independently used agile methods that are suitable in the early project phase for EP. The model is validated using a database of 160 tasks from real agile projects. The prediction accuracy is measured by the percentage of correct predictions among the all predictions. The model results in very good accuracy, but having only one misclassified value. The method achieved 100% prediction in metrics of Pred. (m=25) with 25% tolerance. The MMRE values show that there are no occasional large estimation errors. All other statistical metrics used in this research support these results. In future work, the application can be extended to other deep learning methodologies with higher dimensional data for better performance.

Acknowledgments

The authors would like to thank PES university for allowing to do research and would like to thank Dr. K N B Murthy, vice chancellor, PES university for his encouragement to do research.

References

[1] Y. Shi, M. Li, S. Arndt, and C. Smidts, "Metric-based software reliability prediction approach and its application", *Empirical Software Engineering*, Vol.22, No.4, pp.1579-1633, 2017.

[2] S.K. Dubey and B. Jasra, "Reliability assessment of component based software systems using fuzzy and ANFIS techniques", *International Journal of System Assurance Engineering and Management*, Vol.8, No.2, pp.1319-1326, 2017.

[3] T. Jie, Z. Yong, and W. Lina, "Neural Network Based Software Reliability Prediction with the Feed of Testing Process Knowledge", In: *Proc. of International Conf. On Information Technology and Software Engineering*, Springer, Berlin, Heidelberg, pp.19-27, 2013.

[4] C. Diwaker, P. Tomar, R. C. Poonia, and V. Singh, "Prediction of Software Reliability using Bio Inspired Soft Computing Techniques", *Journal of Medical Systems*, Vol.42, No.5, pp.93, 2018.

[5] N. Khurana, R.S. Chhillar, and U. Chhillar, "A Novel Technique for Generation and Optimization of Test Cases Using Use Case, Sequence", *Activity Diagram and Genetic Algorithm. JSW*, Vol.11, No.3, pp.242-250, 2016.

[6] B.S. Ahmed, "Test case minimization approach using fault detection and combinatorial optimization techniques for configuration-aware structural testing", *Engineering Science and Technology, an International Journal*, Vol.19, No.2, pp.737-753, 2016.

[7] S.R. Sugave, S.H. Patil, and B.E. Reddy, "DDF: Diversity Dragonfly Algorithm for cost-aware test suite minimization approach for software testing", In: *Proc. of International Conf. on Intelligent Computing and Control Systems*, pp.701-707, 2017.

[8] F. Zare, H.K. Zare, and M.S. Fallahnezhad, "Software effort estimation based on the optimal Bayesian belief network", *Applied Soft Computing*, Vol.49, pp.968-980, 2016.

[9] K. Sagar and A. Saha, "A systematic review of software usability studies", *International Journal of Information Technology*, pp.1-24, 2017.

[10] D. E. Strode, "A dependency taxonomy for agile software development projects", *Information Systems Frontiers*, Vol.18, No.1, pp.23-46, 2016.

[11] A.T. Raslan and N.R. Darwish, "An Enhanced Framework for Effort Estimation of Agile Projects", *International Journal of Intelligent Engineering and Systems*, Vol.11, No.3, pp. 205-214, 2018.

[12] S. Bilgaiyan, S. Mishra, and M. Das, "A review of software cost estimation in agile software development using soft computing techniques", In: *Proc. of 2nd International Conf. on Computational Intelligence and Networks*, 2016.

[13] O. Malgonde and K. Chari, "An ensemble-based model for predicting agile software development effort", *Empirical Software Engineering*, pp.1-39, 2018.

[14] P. Xiao, B. Liu, and S. Wang, "Feedback-based integrated prediction: Defect prediction based on feedback from software testing process", *Journal of Systems and Software*, Vol.143, pp.159-171, 2018.

[15] V. Nguyen, B. Boehm, and L. Huang, "Determining Relevant Training Data for Effort Estimation Using Window-based COCOMO Calibration", *Journal of Systems and Software*, Vol.147, pp.124-146, 2018.

[16] Z.W. Zhang, X.Y. Jing, and T.J. Wang, "Label propagation based semi-supervised learning for software defect prediction", *Automated Software Engineering*, Vol.24, No.1, pp.47-69, 2017.

[17] M. Boopathi, R. Sujatha, C.S. Kumar, and S. Narasimman, "Quantification of Software Code Coverage Using Artificial Bee Colony Optimization Based on Markov Approach", *Arabian Journal for Science and Engineering*, Vol.42, No.8, pp.3503-3519, 2017.
[18] S. Bilgaiyan, S. Mishra, and M. Das, "Effort estimation in agile software development using experimental validation of neural network models", *International Journal of Information Technology*, pp.1-5, 2018.
[19] P. Pospieszny, B. Czarnacka-Chrobot, and A. 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.
[20] S. Celar, L. Vickovic, and E. Mudnic, "Evolutionary Measurement Estimation Method for Micro, Small and Medium-Sized Enterprises Based on Estimation Objects", *Advances in Production Engineering & Management*, Vol.7, No.2, pp.81-92, 2012.
[21] S. Čelar, M. Turić, and L. Vicković, "Method for personal capability assessment in agile teams using personal points", In: *Proc. of 22nd International Conf. On Telecommunications Forum Telfor (TELFOR)*, pp.1134-1137, 2014.
[22] J. Hernández-Orallo, P. Flach, and C. Ferri, "A unified view of performance metrics: translating threshold choice into expected classification loss", *Journal of Machine Learning Research*, Vol.13, pp.2813-2869, 2012.
[23] S. T. Kim, S. R. Hong, and C. O. Kim, "Product attribute design using an agent-based simulation of an artificial market*", International Journal of Simulation Modelling*, Vol.13, No.3, pp.288-299, 2014.