AN APPROACH TO ESTIMATE THE DURATION OF SOFTWARE PROJECT THROUGH MACHINE LEARNING TECHNIQUES

Anam Khalid[1], Muhammad Ahsan Latif[1] & Muhammad Adnan[2]

[1]Department of Computer Science, University of Agriculture, Faisalabad, Pakistan
[2]IMIS, Department of computer science & information engineering, NCKU, Taiwan

ABSTRACT

In the software project, to estimate the duration of software processes is frequently a complex problem. Only 39 percent projects are finished on time relative to the original schedule. Many research efforts had been developed to estimate the duration, but no single model could be used which was suitable for this problem. It is a challenging task to recognize a reliable model for estimation. Due to wrong selection for model or assigning weight, a software system faced many problems which lead to cost, time, effort and schedule overrun. This research proposed a procedure to estimate the duration of software projects by applying machine learning technique. The Bayesian regularization back propagation (BR) and Levenberg–Marquardt (LM) training algorithms are used within Feed forward neural network and Radial base neural network and got results of both models. This approach is applied to the data which is taken from the literature review. After training of the models consuming both training algorithms, it is concluded that BR offers superior results than LM.

Keyword: *Duration of Software Project, Artificial Neural Network, Feed Forward Neural Network, Bayesian regularization back propagation*

INTRODUCTION

Software development is a lengthy procedure which comprises numeral development processes and engineering activities. So, ambiguous time period is required to succeed all necessities. Only 39 percent projects are able on time relative to imaginative schedule. Software processes are interleaved arrangements of managerial, technical and cooperative activities with goal of identifying, designing, executing and testing the system (Pendharkar, 2010). The term development refers to three standard stages: design, implementation and testing maintenance deal with expose anomalies, changing operating environment and additional user requirements which demand after delivery of software project (Czerwonka, Nagappan, Schulte & Murphy, 2013). When the software necessities have been quantified, the following categories of predictions must be estimated: how many person-hours are required to develop or retain a software project, how much time is required to develop a software project and duration and how much cost is required for software project which based on resources like tools and people (Gefen, Gefen & Carmel, 2016). Predicting time is vital for software development organizations to manage budget and workload.

Resources requirements can be reformed throughout the implementation of the software project and software growth progression dynamically fluctuates. If project interruptions happen, then it is mandatory to grip this obstacle by recommending an innovative plan. The models which used for prediction must be intelligible and truthful (Ali & Lai, 2016). The techniques and methods used for predicting purpose based on expert judgment, supporting vector machine, multiple linear regressions, Genetic algorithm, COCOMO model and artificial neural network. The neural networks are generally industrialized to estimate and expect the practical and non-practical appearances of software like software delay prediction, problem detection and daily activity prediction (Wang et al., 2016) (Khashei & Bijari, 2010). The contribution of this research is to use feed forward neural network and radial base neural network to estimate the duration of software development project. Bayesian regularization back propagation (BR) and Levenberg–Marquardt (LM) training algorithms are used to train both models. After comparing performances of both approaches through different statistical tactics it is investigated that BR provide superior results than LM.

LITERATURE REVIEW

To build a model which grounded around the requirement history of software development lifecycle and captured essential subtleties of its progress. The logistic regression algorithm had been used with perceived data configuration and inspected features to design the model. This approach leads to satisfied and concrete outcomes (Cerpa, Bardeen, Astudillo & Verner, 2016). This study supposed autonomous necessities and inherent to the only one case study which are the limitations. Some proposed classifiers to predict the outcomes of software projects. Different classifiers were trained to check the performance and conclude that four classifiers (Random Forest, Naive Bayes, Support Vector Machine and Multilayer Perceptrons) perform well. This study had been delivered an online platform by using Random Forest classifier to determine the outcomes of software projects. Some researchers investigates use of Fuzzy Analogy and Classical Analogy with missing data techniques to estimate software development effort (Idri, Abnane & Abran, 2016). The results advocate that Fuzzy Analogy produced more precise performance than a Classical Analogy to estimate effort of software. Furthermore, this study also preferred to use KNN than deletion or toleration because both techniques provide harmful influence on prediction accuracy.

Bisi and Goyal (2016) designed architecture of ANN to predict the software development effort under dependable schedule and financial plan. Logarithmic activation function was used as a supplementary input layer and particle swarm optimization (PSO) was used to train the input values. To condense the dimension of input features principal component analysis (PCA) were applied and in hidden layer to optimize the performance of hidden neurons Genetic Algorithm was applied. The statistical terms MMRE and PRED (25)

conclude that the recommended methodology delivers better results by using, NASA, Albrecht, Desharnais and COCOMO datasets. Gefen (2016) described the Crowdsourcing software development markets (CSMs) elaborate the importance of obtaining information in software development process. At highest degree these markets act as Contract Theory. Their unique size showed possibly unanticipated features of the markets. The limitation of this research is that the data already exist, so novel paradigms of concern cannot be added to it. Lopez and Abran (2015) proposed a radial based function neural network (RBFNN) and multilayer feed forward neural network (MLP) to predict the duration time of new software projects when the team size of developers and functions size are used as the autonomous variables. This research compared with seven other studies and concludes that the proposed technique was statically better than statistical regression method which used to predict the duration of software.

Lopez (2015) used machine learning techniques to solve precision issues of software prediction. Radial Basis Function Neural Network (RBFNN) was compared with feedforward multilayer perceptron (MLP), simple linear regression (SLR) model and general regression neural network (GRNN). Hypothesis test concludes that accurateness gained from RBFNN is statistically better than the SLR, MLP and GRNN when function points are used as the independent variable. Absolute Residuals and Friedman statistical test were used to evaluate the accuracy of the proposed models. It was presented a new method with a combination of analytical programming and use case points method to improve the estimation process of systems. Analytical programming used self-organizing migration algorithm which is very delicate to involve constraints (Urbanek, Prokopová, Silhavy & Sehnálek, 2014). The advantage of this resolution is that, this method had no weight because analytical programming generates weights as coefficients in equations and provides best results to predict the effort. The disadvantage of this method is that the accuracy of Use Case Point method quiet based on User familiarity.

Ozcan and Figlali (2014) attempted to launch an intelligent system to estimate the entire cost of stamping dies. The performance of ANN, multiple regression analysis and conventional approach were examined to estimate the cost of stamping dies. This study exposes that the ANN system performed better for cost estimation as comparatively than linear regression analysis model and conventional approaches. This ANN model was proficient to dropping cost associated uncertainties. Lopez (2013) proposed a feedforward neural network (FFNN) for forecasting the period of new software development projects. FFNN exposed statistically enhanced results than a statistical regression (SR) model when function points are used as the independent variable. Mean Magnitude of Relative Error (MMRE) and an ANOVA statistical investigation test were used to evaluate the accuracy of these two models. At the 90% confidence level FFN produced better outputs than the

SR model to predicting the duration of new software development projects.

Numerous suggested an approach to solve the software development cost estimation (SDCE) problem; this paper proposes an evolutionary morphological approach. The proposed approach uses the dilation erosion perceptron (DEP) with a process, called DEP (MGA), using a modified genetic algorithm (Araujo, Oliveira, Soares & Meira, 2012). A better and more stable global performance of the proposed model, having around 1.86% of averages, improvement regarding the MRLHD, as well as having around 0.64% of average improvement regarding the DEP (BP) is indicated by experimental investigations. Halkjelsvik and Jørgensen (2012) deliver an integrated literature review to forecast the performance of time, which grounded on judgment-based estimates. The main contribution of this research is to condense the imaginable and associating factors through which time forecasting disturbed. This research also compared verified or self-conveyed actual time in the exactness of the time estimates that concern confidence of judgments. Some inspects the effect of process maturity on software development time by originating a new set of "COCOMO II's PMAT" grade values based on Capability Maturity Model Integration (CMMI) (Alyahya, Ahmad & Lee, 2009). Good impression of CMMI maturity procedure of software development time was achieved by exhausting Ideal Scale Factor method (ISF).

MATERIAL AND METHODS

Data

The data is in numeric form and it consists of five inputs and one output value, including Programming language experience, Reused code, New and Changed code, Effort, Team size and Duration. Duration field is taken as an output value. These independent variables are selected according to effort (E) which depends on the percentage of new (NC) and Reusable (RC) code with maximum number of people (TS) that work on the project at any time. Programming language experience (PLE) is also an important point for the selection of the team size. This record is collected from previous study.

Table 1. Sample Data

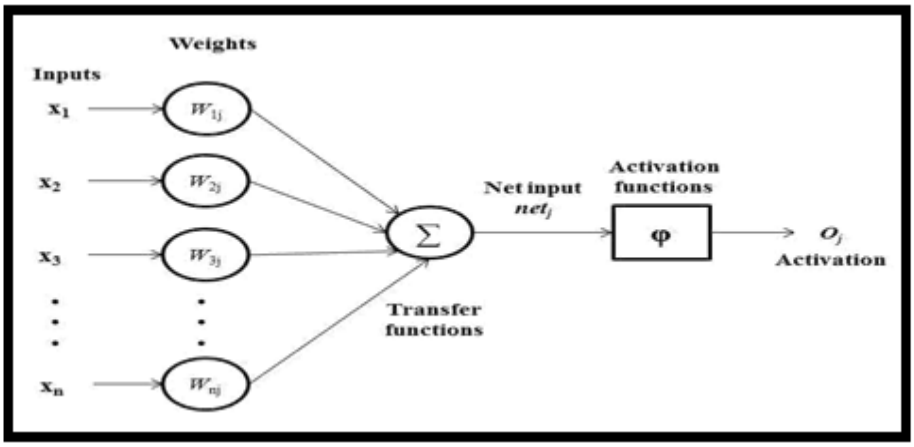
PLE	RC	NC	E	TS	Duration
24	21	17	65	1	68
12	33	11	44	1	62
36	42	10	35	1	53
14	8	95	131	3	153

Artificial Neural Network (NNs)

A neural network (NN) is a network motivated by biological nervous systems and

its structure is based on artificial neurons. The purpose of this network is to estimate or predict functions; these functions can depend on a large number of factors which use numerical values. Each vector is received and observed by neurons with specific independent compassion called weight. The inner condition of the neuron is a summation of the interior merchandise of the input and the weight vectors, and a mathematical assessment called bias. This function which is used to transfer information is called the transfer function (Park & Baek, 2008). Naturally the structures of neural networks are in the form of layers. Input neurons of first layer send data to the second layer of neurons and after updating the weight, data is transferred from second layer to the output layer. Activation functions are placed in these layers that convert neuron input weights to output activation.

Figure 1. Artificial Neural Network



Models

Feed Forward Neural Network

A feedforward neural network was the leading and modest kind of artificial neural network. In this network, the information transfers simply in single track. The signals between neurons always move from the input layer to output layer over hidden nodes. The architecture of FFNN consists of one input layer, one hidden layer with a nonlinear activation function and one output layer with linear combinations of radial basis functions. In neurons the values of weights and bias are adjusted to learn the neural network (Glorot & Bengio, 2010). A FFNN consists of two phases: a training phase and an application phase. During the training phase input values are prepared and constraints are adjusted to improve network performance through learning algorithm. When the training phase is finished and accurate performance is achieved, the network starts its phase of application for the proposed task.

Radial Base Neural Network

A radial basis neural network uses radial basis functions as activation functions. The architecture of RBNN consists of one input layer, one hidden layer with a nonlinear activation function and one output layer with linear combinations of radial basis functions (Nassif, Azzeh, Capretz & Ho, 2016). In this network the neurons value is achieved by adjusting cross product of both bias and weights. To train this network 70% information is required and remaining 30% used for testing and validation purpose. During the training state, constraints are adjusted until required performance is achieved. It is much easier to design and train RBNN than other neural networks.

Algorithm

Levenberg–Marquardt (LM) algorithm

The Levenberg-Marquardt algorithm is a very modest and forceful technique for approaching a function. It reduces the activation functions over a space of constraints and delivers a numerical clarification of the problem. It is suitable for moderate sized problems in ANNs (Jazayeri, Jazayeri & Uysal, 2016). This training algorithm consists of three basic phases: first training phase trains the network, according to the desired output value, second phase validation checked the validity of network and third testing phase test the information. The Levenberg-Marquardt is faster than Bayesian regularization back propagation algorithm, but it is very penetrating to the preliminary weights of the network.

Bayesian regularization back propagation algorithm

Bayesian regularization back propagation algorithm used in neural network to absorb problems and to guesstimate the operative number of constraints essentially required resolving a specific problem. BR modernizes the weights and bias values according to the Levenberg-Marquardt optimization. It reduces the arrangement of weights, squared error, and provides the correct network (Plumb, Rowe, York & Brown, 2005). Overpriced fractious validation can be escaped by exhausting Bayesian regularization. It also eliminates the requirement for testing altered numbers of hidden neurons to solve problem. This training algorithm consists of two phases: training and testing, and it reduces the cost which used in validation phase. Bayesian regularization back propagation slower than Levenberg-Marquardt algorithm, but it is useful to solve complex problems.

Training and Testing the Models

The training, testing and validation procedures of the ANNs are implemented on 360 data permutations, each permutation comprising 5 inputs and 1 target output. Feedforward neural network used the single layer architecture; the activation function in the hidden layer is 10 sigmoid and numbers of neurons are 10. When FFNN was trained through Levenberg-Marquardt algorithm, it passed through three phases: training, validation and testing. The

dividerand function was used to divide the data. This function divides the complete data into the following partitions: 70 percent data used for training, 15 percent data used for testing and 15 percent for validation shown in figure 2.

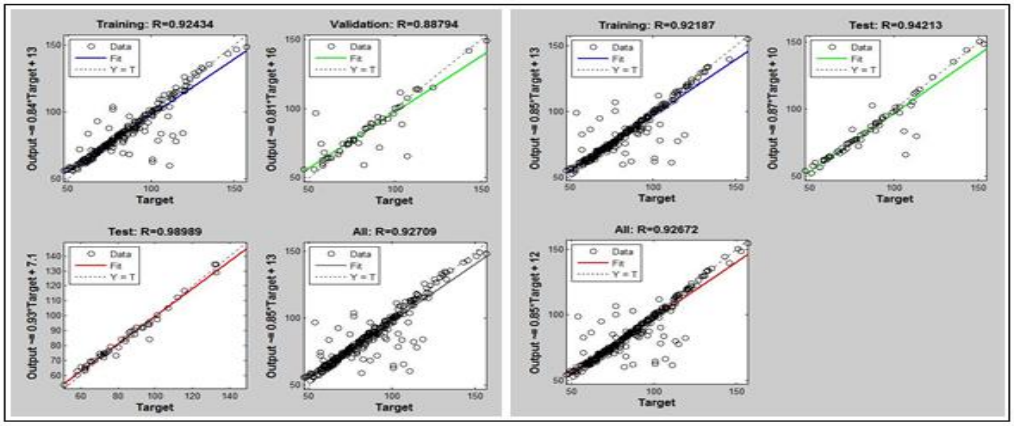


Figure 2. FFNN using LM

Figure 3. FFNN using BR

The identification method is repeated for production of the FFNN with BR training backpropagation algorithm, it passed through two phases: training and testing. The divider and function divides the complete data into the following partitions: 70 percent data used for training and 30 percent data used for testing shown in figure 3. Radial basis neural network also used the single layer architecture. This network used the maximum number of neurons are 40 and the number of neurons to add between displays are 5 and goals set at 0.

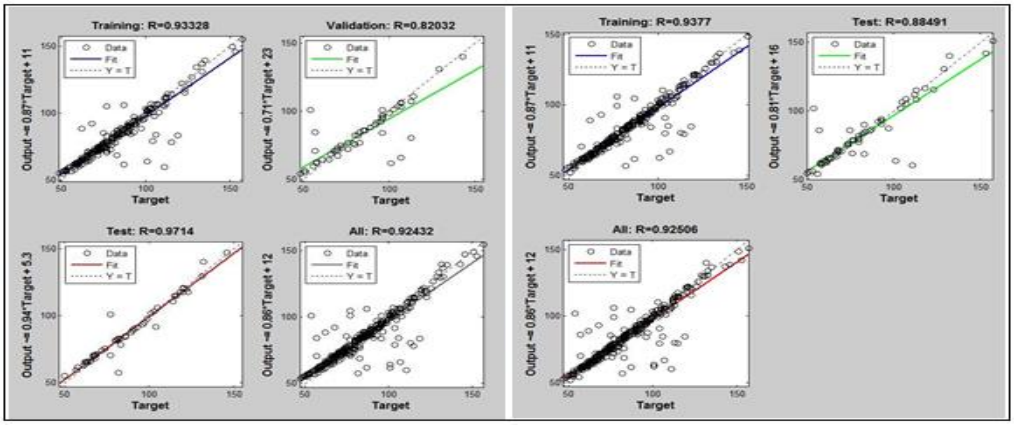


Figure 4. RBNN using LM

Figure 5. RBNN using BR

RBNN also train using both Levenberg-Marquardt and Bayesian regularization back propagation algorithms. Levenberg-Marquardt algorithm passed through three phases:

training, validation and testing, and Bayesian regularization back propagation algorithm passed through two phases: training and testing shown in figure 4 and figure 5.

Modeling Performance Criteria

In order to evaluate the prediction accuracy of FFNN and RBNN through Levenberg-Marquardt and Bayesian regularization back propagation training algorithms, and to find out which algorithm performs well to estimate the duration of software projects, the norms Mean Square Error (MSE) and Coefficient of determination are used. Chi-square test also used to check the normality of the dataset. Mean squared error (MSE) measures the average of the squares of the errors that is the difference between the observed and predicted value. The values of MSE nearer to zero show good results and it is always positive (Franses, 2016). If $\hat{\boldsymbol{Y}}$ is a direction of n forecasts, and $\boldsymbol{Y}$ is the direction of experimental values according to the inputs of the method which produced the predictions, then the MSE can be calculated through Eq:(1):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y} - Y)^2 \qquad \text{Eq:(1)}$$

Coefficient of determination is a number that specifies the percentage of the variation in the dependent variable that can estimate from the independent variable. The coefficient of determination signifies the percentage of the records that is the closest to the best fit line. The answer of Coefficient of determination fall between 0-1 and its maximum value represented good results (Chen & Braga, 2016). It is denoted by R^2.

$$R^2 = \frac{SSR}{TSS} = 1 - \frac{\sum(\hat{Y} - \bar{Y})^2}{\sum(Y - \bar{Y})^2} \qquad \text{Eq:(2)}$$

In the measurements, normality tests are used to conclude that the data set is well-demonstrated by a normal distribution. To check the normality of data sets Chi-square test is used. This test is applied to realize the allocations of defining variables fluctuate from each other (Liu, Tang & Zhang, 2009).

$$X^2 = \sum \frac{(\hat{Y} - Y)^2}{(Y)} \qquad \text{Eq:(3)}$$

Equation (3) denote the chi-square test, $\hat{\boldsymbol{Y}}$ shows the observed values and $\boldsymbol{Y}$ shows the expected values. This test elaborates the normality checks between observed values and expected values. In measurement means a very lesser results from chi square test explain that observed data well fit according to expected data and a relationship exist between data set. A very huge results of the chi - square test means that the fitness of the data is not very well and there is not a relationship between data.

RESULTS AND DISCUSSION

The regression plots of the FFNN for the training, validation and testing processes are displayed in Figure 2 and Figure 3 respectively. Figure 2 demonstrates the performance of FFNN using LM and Figure 3 demonstrates the performance of FFNN using BR. The regression plots of the RBNN for the training, validation and testing processes are given in Figure 4 and Figure 5 respectively. Figure 4 demonstrates the performance of RBNN using LM and Figure 5 demonstrates the performance of RBNN using BR. From the figures it is prominent that LM consists of three phases to train the models, whereas BR comprise of only two phases for this resolution but the performance of both training algorithms are approximately same. Table no. 2 shows the presentation of both algorithms over FFNN and RBNN. This table shows the training, testing and validation outcomes, time period and the number of epochs which is used to train the models.

Table 2. Performance metrics

Performance metrics	**Feedforword Neural network**		**Radial Base Neural network**	
	Levenberg-Marquardt (LM) algorithm	**Bayesian Regularization (BR) algorithm**	**Levenberg-Marquardt algorithm**	**Bayesian Regularization (BR) algorithm**
Best training performance	65.3233	60.2467	65.7809	56.7057
Best validation performance	70.6843	0	34.4626	0
Best testing performance	82.7745	65.0591	85.0380	85.5323
No. of training epochs	14	123	15	113
Best training epoch	8	44	9	109
Training time (in Seconds)	01	13	10	26

The results of these networks, elaborate that BR used less training and testing performance than LM and LM used less epochs and time as compared than BR. But the validation in BR is zero. The Less value of performances illustrate better results. To check the performances of both algorithms, these two networks are run five times and get results in the form of Mean Square Error (MSE) and Coefficient of determination (R^2). Table no. 3 shows the values of both models after applying both algorithms.

Table 3. Statistical Test

Statistical Test	**Feedforword Neural network**		**Radial Base Neural network**	
	LM	**BR**	**LM**	**BR**
MSE	63.9202	60.9207	63.1766	59.1262
	63.6700	60.2009	67.3238	61.5580
	73.9866	61.1689	61.3707	61.1453
	66.4048	61.5746	60.9565	62.5065

	61.1533	60.9045	67.9373	62.3673
R^2	0.85221	0.85883	0.85883	0.86291
	0.85259	0.86479	0.86048	0.85751
	0.85703	0.85829	0.85829	0.85855
	0.86083	0.85727	0.85727	0.85544
	0.85831	0.85888	0.85888	0.85542

The less outcome value of MSE deliberate better results. After training of both models, following results are conducted: the average values of MSE provide less value and R^2 provide greater value while using BR.

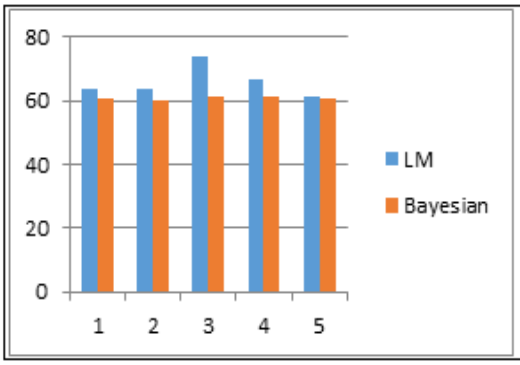


Figure 6. MSE of FFNN

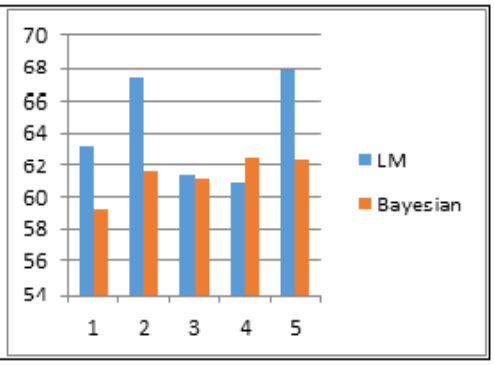


Figure 7. MSE of RBNN

Figure no. 6 reveal the results in MSE to evaluate the performance of both algorithms for FFNN and Figure no.7 expose the performance of both algorithms for RBNN. Figure no. 8 and Figure no. 9 elaborate the R^2 values of both models.

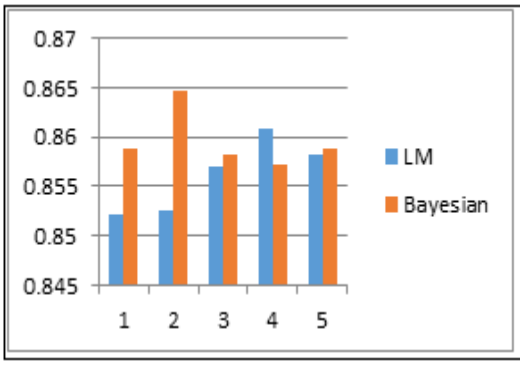


Figure 8. R^2 of FFNN

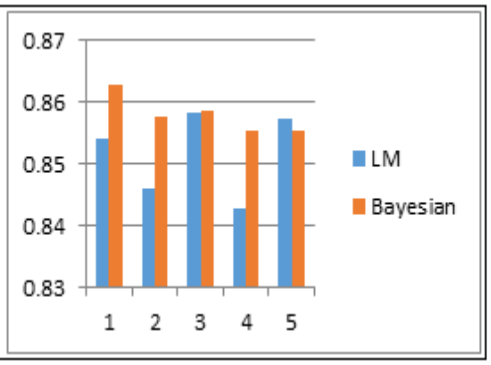


Figure 9. R^2 of RBNN

Chi-squared test is used to check the normality of datasets through p-value. Normality test is compulsory to check that the dataset is in formal form or not. Table no. 3 shows the normality of the dataset which conducting from both models at 95% confidence interval. P-values less than 0.05 shows good results and it's elaborate that dataset is in formal form.

Chi-squared test is realistic between the dataset and the results which are obtained after applying both algorithms.

Table 3. Normality Test

Chi-Square Tests				
Model	**Algorithm**	**Value**	**Df**	**P-Value**
FFNN	BR	28428.331[a]	25800	.0005
	LM	28428.331[a]	25800	.0002
RBNN	BR	28428.331[a]	25800	.0001
	LM	28428.331[a]	25800	.0050

The results of Chi-squared test, demonstrate that the relationship between data and results is in formal form of both models. The p-values of BR and LM are less than 0.05.

CONCLUSION

To estimate the duration of a software project is most precious and important part of the software development phase. Duration time means the time, which is used to develop a software project. Two machine learning models are realistic to estimate the duration time by practicing two different training algorithms. FFNN and RBNN both models are trained through Levenberg–Marquardt (LM) algorithm which consume three variables (train, test and valid) and Bayesian regularization back propagation algorithm which consume only two variables (train and test), it avoid validation process. The performances of these algorithms are compared through MSE and R^2 statistical tool and accomplish that BR deliver slightly superior outcomes to estimate the duration of software than LM. BR is more preferred for this determination since it avoids the validation procedure which is cost effective.

References

Ali, N., & Lai, R. (2016). A method of requirements change management for global software development. *Information and Software Technology, 70*:49-67.

Alyahya, M. A., Ahmad, R., & Lee, S. P. (2009). Effect of CMMI-based software process maturity on software schedule estimation. *Malaysian Journal of Computer Science, 22* (2):121-137.

Araujo, R. D. A., Oliveira, A. L., Soares, S., & Meira, S. (2012). An evolutionary morphological approach for software development cost estimation. *Neural Networks*, *32*: 285-291.

Bisi, M., & Goyal, N. K. (2016). Software development efforts prediction using artificial neural network. *IET Software, 10* (3):63-71.

Cerpa, N., Bardeen, M., Astudillo, C. A., & Verner, J. (2016). Evaluating different families of prediction methods for estimating software project outcomes. *Journal of Systems and Software*, *112*:48-64.

Chen, T., & Braga-Neto, U. M. (2016). Bayesian estimation of the discrete coefficient of determination. *EURASIP Journal on Bioinformatics and Systems Biology*, (1): 1.

Czerwonka, J., Nagappan, N., Schulte, W., & Murphy, B. (2013). Codemine: Building a software development data analytics platform at Microsoft. *IEEE software*, *30* (4):64-71.

Franses, P. H. (2016). A note on the mean absolute scaled error. *International Journal of Forecasting*, *32* (1): 20-22.

Gefen, D., Gefen, G., & Carmel, E. (2016). How project description length and expected duration affect bidding and project success in crowdsourcing software development. *Journal of Systems and Software*, *116*:75-84.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*: 249-256.

Halkjelsvik, T., & Jørgensen, M. (2012). From origami to software development: A review of studies on judgment-based predictions of performance time. *Psychological bulletin*, *138* (2):238.

Idri, A., Abnane, I., & Abran, A. (2016). Missing data techniques in analogy-based software development effort estimation. *Journal of Systems and Software*, *117*:595-611.

Jazayeri, K., Jazayeri, M., & Uysal, S. (2016, July). Comparative Analysis of Levenberg-Marquardt and Bayesian Regularization Backpropagation Algorithms in Photovoltaic Power Estimation Using Artificial Neural Network. In *Industrial Conference on Data Mining*: 80-95.

Khashei, M., & Bijari, M. (2010). An artificial neural network (p, d, q) model for timeseries forecasting. *Expert Systems with applications*, *37* (1): 479-489.

Liu, H., Tang, Y., & Zhang, H. H. (2009). A new chi-square approximation to the distribution of non-negative definite quadratic forms in non-central normal variables. *Computational Statistics & Data Analysis*, *53* (4):853-856.

Lopez-Martín, C. (2015). Predictive accuracy comparison between neural networks and statistical regression for development effort of software projects. *Applied Soft Computing*, *27*:434-449.

López-Martín, C., & Abran, A. (2015). Neural networks for predicting the duration of new software projects. *Journal of Systems and Software*, *101*:127-135.

Lopez-Martin, C., Chavoya, A., & Meda-Campaña, M. E. (2013). Use of a feedforward neural network for predicting the development duration of software projects. In *Machine Learning and Applications, 12th International Conference*, 2: 156-159.

Nassif, A. B., Azzeh, M., Capretz, L. F., & Ho, D. (2016). Neural network models for software development effort estimation: a comparative study. *Neural Computing and Applications*, *27* (8):2369-2381.

Ozcan, B., & Fıglali, A. (2014). Artificial neural networks for the cost estimation of stamping dies. *Neural Computing and Applications*, *25* (3-4):717-726.

Park, H., & Baek, S. (2008). An empirical validation of a neural network model for software effort estimation. *Expert Systems with Applications*, *35* (3):929-937.

Pendharkar, P. C. (2010). Probabilistic estimation of software size and effort. *Expert Systems with Applications*, *37* (6):4435-4440.

Plumb, A. P., Rowe, R. C., York, P., & Brown, M. (2005). Optimization of the predictive ability of artificial neural network models: A comparison of three ANN programs and four classes of training algorithm. *European Journal of Pharmaceutical Sciences*, *25* (4):395-405.

Urbanek, T., Prokopová, Z., Silhavy, R., & Sehnálek, S. (2014). Using analytical programming and UCP method for effort estimation. In *Modern Trends and Techniques in Computer Science*: 571-581.