ISSN: 2277-9655
Impact Factor: 4.116

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

EMPIRICAL VALIDATION OF RANDOM FOREST FOR AGILE SOFTWARE EFFORT ESTIMATION BASED ON STORY POINTS

Anjali Sharma*, Karambir
* Student of Department of CSE, UIET, KUK, India
Assistant Professor, Department of CSE, UIET, KUK, India

DOI: 10.5281/zenodo.58604

ABSTRACT

Agile Software development has become famous in industries and replacing the traditional methods of software development. A correct estimation of effort in this concept still remains an argument in industries. Thus, the industry must be able to estimate the effort necessary for software development using agile methodology. For estimating effort different types of neural-networks Probabilistic Neural Network (PNN), General Regression Neural-Network (GRNN), Group Method of Data Handling (GMDH) Polynomial Neural-Network and Cascade-Correlation Neural-Network) are used. To achieve better prediction, effort estimation of agile projects researchers used Random Forest with Story Points Approach (SPA) in the place of neural-network because Random Forest is easy to implement and better than decision tree. Random Forest gives better results as compare to neural-network.

KEYWORDS: Agile, Agile Software Development, Neural-Network, Software Effort Estimation, Story Point approach, Random Forest.

INTRODUCTION

Agile methodologies are used to develop and implement software quickly according to customer requirements. Agile Software Development Methodologies share some of the features including iterative development, prototyping and the minimal documentation. Agile software development methodologies are applied to create the high quality software in the shorter period of time. It is a substitute of the traditional project management used in software development. Agile software development is a methodology for creative process that waits for the need for flexibility and applies a level of practicality into the delivery of the complete product. Agile methods are used for developing software to allow organizations respond to volatility. They provide chances to assess the direction all through the software development life cycle [1]. By accenting on the repetition of work cycles along with product the teams return an additive and iterative development. Instead of the promising to market an assemble software that hasn't been developed, agile allows teams to frequently re-plan their release to optimize its value throughout development in the marketplace making them competitor [2] [3]. Predictability is the main goal of project management, we require to be able to estimate the size and complexity of the products to be built in order to decide what to do next [4]. For this, requirements need to be collected. Requirements in agile development are counted down in cards and are called user stories. These stories are estimated using story points. The team explains the relationship between story point and effort. Generally 1 story point is equal to 1 ideal working day. Total no. of story points that a team can convey in a sprint (an iteration in agile software development) is called as “team velocity” or story points per sprint. Now for obtaining better prediction accuracy, Random Forest Method is used in this study. The results found by applying this method is empirically validated and compared to measure their performance.

Software effort estimating is an important but difficult task. Software effort estimation process in any software project is not only essential, but also a very critical component. The success or failure of projects depends heavily on the accuracy of effort and schedule estimations, among other things [5]. In this paper a software effort estimation model for Agile Software projects has been presented. The model uses User Stories of as base for estimation. In order to address different challenges faced by the agilest, the model is developed to contain most of the characteristics of agile

methodology, especially Adaption and Iteration. In this Random Forest has been presented with story point approach to predict better results as compare to neural networks.

Cost Estimation Techniques

Cost estimation tools, or model-based estimation techniques use data collected from past projects combined with mathematical formula to estimate project cost. These models need system size as input. The main model-based techniques include COCOMO, SLIM, RCA PRICE-S, SEER-SEM, and ESTIMACS. The existing effort estimation techniques are generally classified as regression-based models, learning-oriented models, composite-Bayesian methods and expert based approaches.

Most of the software estimation models are based on regression technique [6]. Regression models generally use previous data, created by collecting data on completed projects and developing regression equations that describe the relationships among the different variables. Estimates are made by substituting the new project parameters are substituted into mathematical model. This model is evaluated on regression data to make estimates. In these models software development effort is only dependent variable of some predicted variables like Effort adjustment factors, Size, Scaling factors etc. for regression equation.

Regression models need certain conditions in some cases to be fulfilled particularly. These conditions are examined by Boehm and Sullivan, and are based on experience from the use of regression-based models. These typical conditions include accessibility of a large dataset, no outliers, no missing data items, and the predictor variables are not correlated. The collection of approaches that fall under the heading of regression-models include classification and regression trees (CART), ordinary least-squares regression (OLS), stepwise analysis of variance for unbalanced data sets (stepwise ANOVA), combinations of CART with OLS regression and analogy, multiple linear regression, and stepwise regression [7].

There are other types of model, called Learning-oriented models which are based on learning from previous estimation experience. These models attempt to automate the estimation process by training themselves from previous experience to build computerized models [8]. These models are capable of learning incrementally and refining themselves as new data are offered over time. Learning-oriented models cover a wide area and include techniques such as artificial neural networks artificial intelligence approaches, case-based reasoning, machine learning models, decision-tree learning, fuzzy logic models, knowledge acquisition and rule induction[9]. The main model-based techniques include COCOMO, SLIM, RCA PRICE-S, SEER-SEM, and ESTIMACS. These estimation models produce an estimate of the cost, effort or duration of a project based on factors such as the size and desired functionality of the system.

An important expertise based approach was found by Briand et al. (1998) to be “comparison to similar, past projects based on personal memory”. The expertise based approaches are useful when no quantified, empirical data is available. They provide a practical, low-cost and highly useful process. Another estimation technique used for software effort estimation is analogy based estimation. The technique studies past projects and uses the information retrieved as a guide estimate for the proposed project. The Checkpoint method is an example of an analogy-based approach to software estimation. In this technique heuristics are derived from actual project data or a formalization of expert opinion. In order to derive these heuristics some form of project data or information are used. These heuristics are, used to estimate productivity, quality or size. Expert judgment Estimation is one of the popular estimation techniques in software effort estimation which is based on the gathered experiences of teams of experts in order to come up with project estimates. This technique is used where the estimation process is mainly based on “non-explicit, non-recoverable reasoning processes”.

Expert Judgment techniques have been criticized by experts for their dependence on human memory and the lack of repeatability of such memory-based approaches [10] [11] however reports have proven it to be the dominant strategy in software development estimation. The Delphi technique and work breakdown structure (WBS), top-down and bottom-up estimation reasoning by analogy, formal reasoning by analogy, informal reasoning by analogy, and rules of thumb [12] fall under expert judgment technique.

The strengths of expertise based methods and regression-based methods were combined to introduce a new estimation approach called the Bayesian approach which is a semi-formal estimation process [13]. Bayesian analysis allows for the fact that the data required for use in most estimation techniques is usually of poor quality or incomplete. Expert judgment is incorporated in this approach to handle the missing data and provide a more robust estimation process [8, 14]. Bayesian analysis has been used in many scientific disciplines and was used in the development of the COCOMO II model [15]. Cost Estimation, Benchmarking and Risk Analysis (COBRA) is an example of a composite estimation model [16].

Effort Estimation practice in agile software Development

In waterfall a team member's workload capacity is decided by the manager who estimates how long certain tasks will obtain and then assigns work based on that team member's total accessible time. Agile methodology takes an extensively different approach to determining a team member's capacity. First of all, it assigns work to a whole team, not an individual. Philosophically, this places a pressure on collective effort. Second, it refuses to quantify work in terms of time because this would fail the self-organization central to the success of methodology. This is a main break from waterfall: Instead of a manager estimating time on behalf of other individuals and assigning tasks based on assumption, team members in Scrum use effort and degree of difficulty to estimate their own work.

Agile Methodology does not advise a single way for teams to estimate their work. However, it does ask that teams not estimate in terms of time, but, instead, use a more abstracted metric to quantify effort. Common estimating methods include numeric sizing, t-shirt sizes, the Fibonacci sequence and even dog breeds. The important thing is that the team shares an understanding of the scale it is uses, so that every member of the team is comfortable with the scale's values. In the Sprint Planning Meeting, the team sits down to estimate its effort for the stories in the backlog. The Product Owner needs these estimates, so that he or she is empowered to effectively prioritize items in the backlog and, as a result, forecast releases based on the team's velocity. This means the Product Owner needs an honest appraisal of how difficult work will be. Thus it is recommended that the Product Owner does not observe the estimation process to avoid pressuring a team to reduce its effort estimates and take on more work. Even when the team estimates amongst itself, actions should be taken to reduce influencing how a team estimates. As such, it is recommended that all team members disclose their estimates simultaneously. Because individuals "show their hands" at once, this process is like a game of poker.

Still, even when teams possess a shared understanding of their scale, they can't help but estimate differently. To arrive at a single effort estimation that reflects the entire team's sense of a story's difficulty, it often requires numerous rounds of estimation. Veteran teams who are familiar with the process, however, should reach a consensus after just a few rounds of planning poker.

PROPOSED WORK

The proposed Model is implementing using the twenty one project data set developed by six software houses. The data set is three-dimensional. The first dimension indicates the number story points required to complete the project, the second represents the velocity of the project, and the third represents the actual effort required to complete that project. This data is used by (Ziaudden *et al.*) for predicting effort using regression. In this study, in order to enhance the effort estimation accuracy, random forest is employed.

Most of the Software Effort Estimation Models estimate Cost, Duration and Personnel for a project. But it will not be the case for Agile Development. There are several key differences between the agile approach to team organization and the traditional approach.

The steps taken to determine the effort of a software product are described below:

- Collection of total number of Story Points: Project velocity and actual effort. The total number of storypoints, project velocity values and actual effort are collected from (Ziaudden *et al.*).
- Normalization of Data Set: This step deals with generating the normalized values of the total number of story points and project velocity within the range [0,1].

DETERMINING THE EFFORT

There is a multitude of factors that affect our ability to accurately estimate effort. Accurate estimation requires a multidimensional view to produce accurate and effective estimates. The challenge, however, is which dimensions do we measure? If we were to classify the possibilities using a SWOT according to Internal vs. External influences, we can eliminate many of the candidates by simply focusing our attention on the things over which we have influence and conversely paying less attention to those that we can't. We keep the vectors to two so as to keep the process as simple as possible so that we actually use the process and don't try to sidestep it because it is too cumbersome. Using two vectors also maintains a consistency with the other areas of the methodology.

Story Size

Story size is an estimate of the relative scale of the work in terms of actual development effort. Table 1 shows five values, assigned to different types of user stories according to their size. Wording of the Guideline description can be changed by the Team itself or even the criteria can be redefined.

Table 1. Story Size Scales

Value	Guidelines
5	• An extremely large story • Too large to accurately estimate • Should almost certainly be broken down into a set of smaller Stories • May be a candidate for separation into a new project
4	• A very large Story • Requires the focused effort of a developer for a long period of time – Think in terms of more than a week of work • Should consider breaking it down into a set of smaller stories
3	• A moderately large story • Think in terms of two to five days of work
2	• Think in terms of a roughly a day or two of work
1	• A very small story representing tiny effort level. • Think in terms of only a few hours of work.

Complexity

This is complexity of either or both the requirements of the Story and or its technical complexity. Complexity introduces uncertainty to the estimate – more complexity means more uncertainty. Table 2 shows 5 values, assigned to user stories according to their nature. Like Story Size table, these guidelines are not fixed. These can be adjusted by the team itself; however we have categorized them to accommodate all characteristics of Agile software development methodology.

Table 2. User Story Complexity Scale.

Value	Guidelines
5	• Extremely complex • Many dependencies on other stories, other systems or subsystems • Represents a skill set or experience that is important, but absent in the team • Story is difficult to accurately describe • Many unknowns • Requires significant refactoring • Requires extensive research • Requires difficult judgment calls • Effects of the Story have significant impact external to the story itself.
	• Very complex • Multiple dependencies on other stories, other systems or subsystems • Represents a skill set or experience that is important, but not strong in the team • Story is somewhat difficult for product owner to accurately describe

ISSN: 2277-9655
Impact Factor: 4.116

4	• Multiple unknowns • Comparatively large amount of refactoring required • Requires research • Requires senior level programming skills to complete • Requires somewhat difficult judgment calls • Effects of the Story have moderate impact external to the story itself
3	• Moderately complex • Moderate number of dependencies on other stories, other systems or subsystems • Represents a skill set or experience that is reasonably strong in the team • Story is somewhat difficult for owner to accurately describe • Moderate level of unknowns • Some refactoring may be required • Requires intermediate programming skills to complete • Requires little research • Requires few important judgment calls • Effects of the Story have minimal impact external to the story itself
2	• Easily understood technical and business requirements • Little or no research required • Few unknowns • Little if any research required • Requires basic to intermediate programming skills to complete • Effects of the Story are almost completely localized to the Story itself
1	• Very straightforward with few if any unknowns • Technical and business requirements very clear with no ambiguity • No unknowns • No research required • Requires basic programming skills to complete • Effects of Story are completely localized to the Story itself

Using these two vectors, effort of a particular User Story is determined using the following simple formula:

ES= Complexity x Size

Effort for the complete project will be sum of efforts of all individual user stories.

$$\mathbf{E} = \sum_{i=1}^{n} (\mathbf{ES})_i$$

The unit of Effort is Story Point (SP). A Story Point is the amount of effort, completed in a unit time.

EXPERIMENTAL DETAILS

For implementing the proposed Model, the data set given in (Ziaudden *et al.*) is used. The detailed description about the data set has been given in the proposed Model. The inputs to the random forest models are total number of story points and project final velocity and the output is the effort i.e., the completion time and Cost. The model is tested and validated using leave-one-out validation for achieving better accuracy.

Story Point

Story Point approach is a most popular approach of calculating effort of agile projects mathematically. Story Point is a metric used in the agile software development to estimate the effort to implement a story. A story point is a particular business need assigned to the software development team. Using estimations of story points rather than time allows development teams to be less precise. In simple terms it is a number that tells the team how hard the story is. Hard could be related to complexity and effort. It is a relative term and does not co-relate to the actual hours. Story Point have no relevance to actual hours, it makes it easy for scrum teams to think abstract about the effort required to complete a story.

Random Forest

Random forest is a notion of general technique of random decision forests that are an ensemble learning method for classification, regression and other tasks, that operate by making a multitude of decision trees at training time and outputting the class that is mode of the classes or mean prediction (regression) of the individual trees. Random decision forests exact for the decision trees habit of over fitting to their training set. The selection of a random subset of features is an example of the random sub-space method which, in Ho's formulation, is a way to implement "stochastic discrimination" approach to classification proposed by Eugene Kleinberg. General method of the random decision forests was 1st proposed by Ho in 1995. Random forest is an ensemble classifier that will consists of many decision trees and outputs the class that is the mode of the class's output by individual trees. The method merges Breiman's "Bagging" idea and the random selection of features.

Random Forest algorithm works as a large collection of decorrelated decision tree. Forest means a lot of decision trees are used. Random Forest is an ensemble approach that can also be thought of as a form of Nearest Neighbor Predictor. It is an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees. Ensembles are divide and conquer approach used to improve the performance. The main Principle behind ensemble method is that a group of weak learners can come together to form a strong learner. Each classifier individually is a weak learner while all the classifiers taken together are a strong learner.

Algorithm

Each tree is constructed using the following algorithm:

1. Let the number of training cases be *N*, and the number of variables in the classifier be *M*.
2. We are told the number *m* of input variables to be used to determine the decision at a node of the tree; *m* should be much less than *M*.
3. Choose a training set for this tree by choosing *n* times with replacement from all *N* available training cases (i.e. take a bootstrap sample). Use the rest of the cases to estimate the error of the tree, by predicting their classes.
4. For each node of the tree, randomly choose *m* variables on which to base the decision at that node. Calculate the best split based on these *m* variables in the training set.
5. Each tree is fully grown and not pruned (as may be done in constructing a normal tree classifier).

For prediction a new sample is pushed down the tree. It is assigned the label of the training sample in the terminal node it ends up in. This procedure is iterated over all trees in the ensemble, and the average vote of all trees is reported as random forest prediction.

RESULTS

For implementing the proposed approach (Random Forest), the data set given in the (Zia *et al.*) is used. The inputs to the Random Forest are the total number of the story points and the output is the effort i.e. completion time. Random Forest is tested and validated for achieving better accuracy.

Comparison of MSE

Fig 1 demonstrates the comparison mean Square Error (MSE) values for different types of Neural Network (GRNN, PNN, GMDH, CCNN) and Random Forest. Among all types of Models, Random Forest performs better. The learning process in Random Forest is quick. The Implementation is easy as compare to neural networks and also performs better than decision tree.

ISSN: 2277-9655
Impact Factor: 4.116

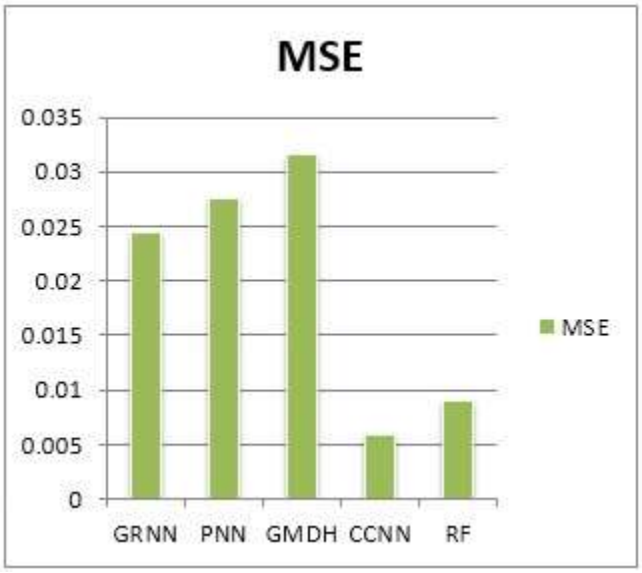

Figure 1. Comparison of MSE

Comparison of R^2

Fig 2 demonstrates the comparison Squared Correlation Coefficient (R2) values for different types of Neural Network (GRNN, PNN, GMDH, CCNN) and Random Forest. Among all types of Models, Random Forest gives better value of R^2.

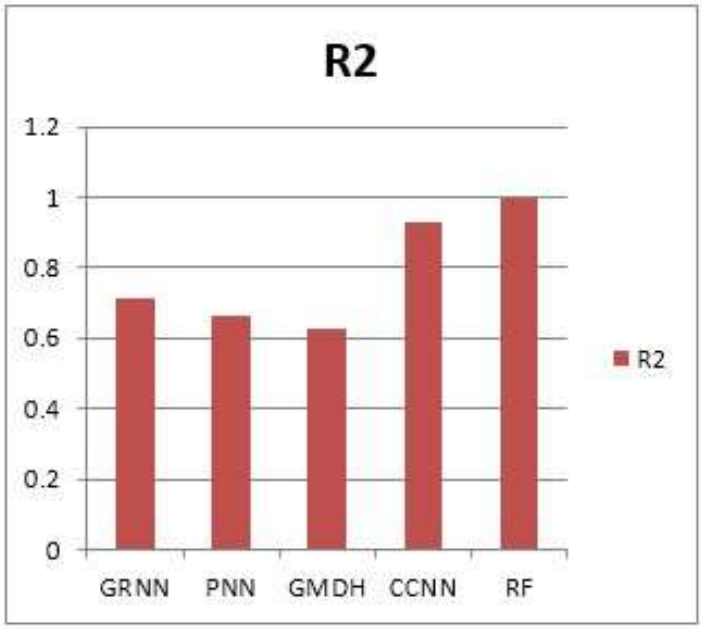

Figure 2.Comparison of R^2

Comparison of MMRE

Fig 3 demonstrates the comparison Mean Magnitude of Relative Error (MMRE) values for different types of Neural Network (GRNN, PNN, GMDH, CCNN) and Random Forest. Among all types of Models, Random Forest gives better value of MMRE.

ISSN: 2277-9655
Impact Factor: 4.116

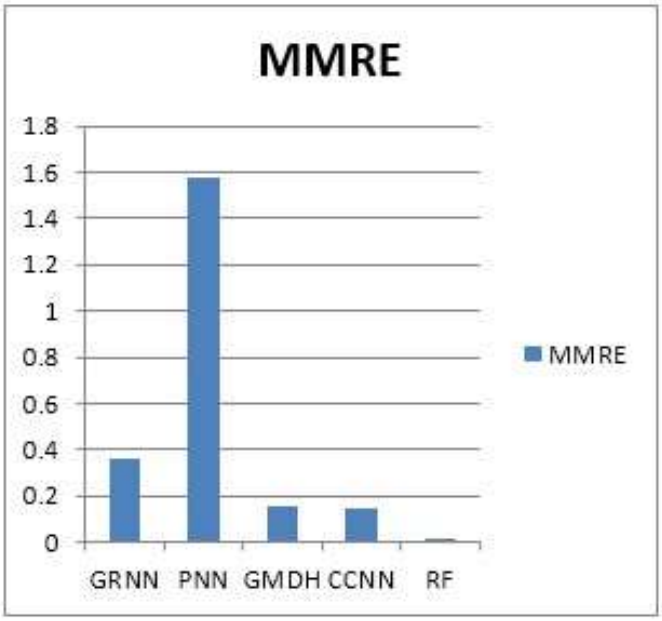

Figure 3 Comparison of MMRE

Comparison of PRED

Fig 4 demonstrates the comparison Prediction Accuracy (PRED) values for different types of Neural Network (GRNN, PNN, GMDH, CCNN) and Random Forest. Among all types of Models, Random Forest provides better accuracy.

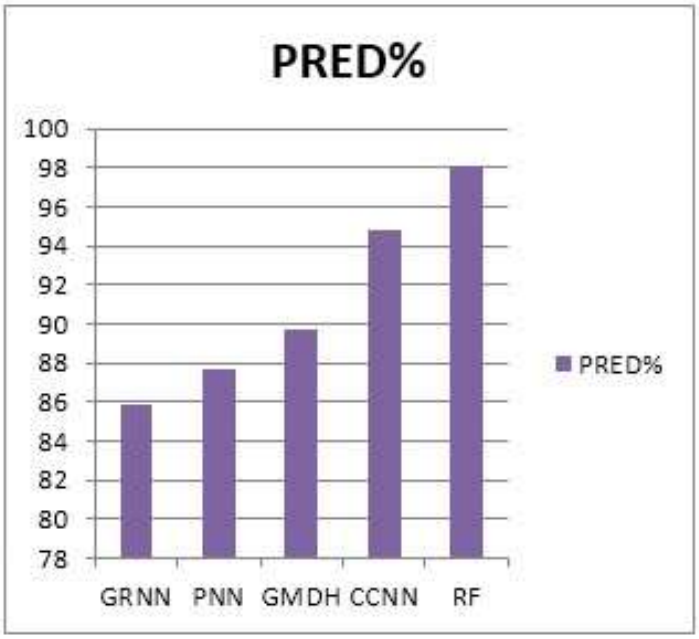

Figure 4. Comparison of Prediction Accuracy

PERFORMANCE METRICS

A **performance metric** is that which decides an organization's behavior and performance. Performance metrics measured the organization activities and performance.

ISSN: 2277-9655
Impact Factor: 4.116

- **MSE**

The **mean squared error** (**MSE**) or **mean squared deviation** (**MSD**) of an estimator measures the average of the squares of the errors or deviations, means the difference between the estimator and what is estimated.
The Mean Square Error (MSE) is calculated:

$$\mathbf{MSE} = \sum_{\mathbf{i=1}}^{\mathbf{TD}} (\mathbf{AE}_i - \mathbf{PE}_i)^{\,2} / \mathbf{TD}$$

Where AE_i = Actual Effort of i test data and PE_i = Predicted Effort of i test data and TD = Total Number of Data.

- **MMRE**

The Mean Magnitude of Relative Error, MMRE, is probably the most widely used evaluation criterion for assessing the performance of competing software prediction models.
The Mean Magnitude of Relative Error (MMRE) is calculated :

$$\sum_{\mathbf{i=1}}^{\mathbf{TD}} \left(\left| \mathbf{AE}_i - \mathbf{PE}_i \right| \, / \, \mathbf{AE}_i \right)$$

- **Squared Correlation Coefficient (R2)**

It is a statistic used in the context of statistical models whose main purpose is either the prediction of future outcomes.
The squared correlation coefficient (R2) is calculated as:-

$$\mathbf{R}^2 = \mathbf{1} - \sum_{\mathbf{i=1}}^{\mathbf{TD}} (\mathbf{AE}_i - \mathbf{PE}_i)^2 / \left(\sum_{\mathbf{i=1}}^{\mathbf{TD}} \mathbf{AE}_i - \mathbf{AE} \right)$$

- **Prediction Accuracy (PRED)**

It is a description of systematic errors and random errors.
The Prediction Accuracy (PRED) is calculated as:

$$\mathbf{PRED} = \left(\mathbf{1} - \left(\sum_{\mathbf{i=1}}^{\mathbf{TD}} \left(\left| \mathbf{AE}_i - \mathbf{PE}_i \right| \right) / \mathbf{TD} \right) \right) * \mathbf{100}$$

CONCLUSION

In this paper Random Forest and Story Point approach was used for estimation the effort of a real life case study. Random forest is a notion of general technique of random decision forests that are an ensemble learning method for classification, regression and other tasks. Story point approach is one of the method that can be used for developing mathematical models for agile software effort estimation. At the end of this paper results obtained from Random Forest and all types of Neural Network (GRNN, PNN, GMDH and CNN). The comparison of both was shown in the table and graph. As shown in this paper random forest gives better results as compare to all types of Neural Network (GRNN, PNN, GMDH and CNN). The computations for above methodologies were executed, and results were obtained using MATLAB.

The proposed work contains dataset of twenty one records but it can be increase for the further study purposes. For the future work on this field the large size of dataset should be available for better performance.

REFERENCES

[1] Martin Fowler and Jim Highsmith. The agile manifesto, “Software Development. San Francisco, CA: Miller Freeman, Inc”, pp. 28-35, 2001.
[2] David Cohen and Mikael Lindvall and Patricia Costa, “An introduction to agile methods. Advances in Computers”, Elsevier, pp. 1-66, 2003.
[3] Rashmi Popli and Naresh Chauhan. Estimation in agile environment using resistance factors. Information Systems and Computer Networks (ISCON), International Conference, IEEE, pp. 60-65, 2014.
[4] Shashank Mouli Satapathy, Mukesh Kumar and Santanu Kumar Rath. Fuzzy-class point approach for software effort estimation using various adaptive regression methods. CSI Transactions on ICT, Springer, pp. 367-380, 2013.

[5] Zia, Z.; Rashid, A.; Uzzaman, K., "Software cost estimation for component based fourth-generation-language software applications, IET Software", vol. 5, pp. 103-110, 2001.
[6] Matson, J. E., Barrett, B. E. & Mellichamp, J. M., " Software Development Cost Estimation Using Function Points", IEEE Transactions on Software Engineering, vol. 20**,** pp. 275-287, 1994.
[7] Keaveney S. and Conboy K., "Cost Estimation in Agile Development Projects", Proceedings of the 14th European Conf. Information Systems (ECIS), 2006.
[8] Boehm, B. W., ABTS, C. and Chulani S., "Software Development Cost Estimation Approaches: A Survey. USC-CSE", 2000.
[9] Burgess, C. J. and Lefley M., "Can Genetic Programming Improve Software Effort Estimation? A Comparative Evaluation. Information and Software Technology", Vol. 43, pp. 863-873, 2001.
[10] Briand, L. C., El emam, K. and Bomarius, F.," COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking, and Risk Assessment", Proceedings of the 20th International Conference on Software Engineering. Kyoto, Japan, 1998.
[11] Mukhopadhyay, T. and Kekre, S.," Software Effort Models for Early Estimation of Process Control Applications", IEEE Transactions on Software Engineering, Vol. 18**,** pp. 915-924,1992.
[12] Mendes, E., Watson, I., Triggs, C., Mosley, N. and Counsell, S.," A Comparison of Development Effort Estimation Techniques for Web Hypermedia Applications", Proceedings of the 8th IEEE Symposium on Software Metrics,2002.
[13] Jones, C.," By Popular Demand: Software Estimating Rules of Thumb. Computer, Vol. 29**,** pp. 116-118, 1996.
[14] Ferens, D. V.," Software Size Estimation Techniques. Proceedings of the IEEE National Aerospace and Electronics Conference", 1991.
[15] Boehm, B. W. and Sullivan, K. J., "Software Economics: Status and Prospects. Information and Software Technology", Vol. 41, 937-946, 1991.
[16] Ruhe, M., Jeffery, R. and Wieczorek, I.," Cost Estimating for Web Applications", Proceedings of the 25th International Conference on Software Engineering. Portland, Oregon, 2003.