Software effort prediction and classification by optimizing features with Random forest Approach by Boosting

Rupsi Bedi[1], Monika Pathania [2]
[1, 2] *Computer science, Bells Institute of Management and Technology, Shimla, INDIA*

Abstract- Analogy-based software effort estimation is one of the most popular estimation methods. It is built upon the principle of case-based reasoning (CBR) based on the *k*-the similar projects completed in the past. Therefore the determination of the *k* value is crucial to the prediction performance. Various research have been carried out to use a single and fixed *k* value for experiments, and it is known that dynamically allocated *k* values in an experiment will produce the optimized performance. This paper proposes an interesting technique based on hierarchical clustering to produce a range for *k* through various cluster quality criteria. We find that complete linkage clustering is more suitable for large datasets while single linkage clustering is suitable for small datasets. The method searches for optimized *k* values based on the proposed heuristic optimization technique, which have the advantages of easy computation and optimized for the dataset being investigated. Datasets from the PROMISE repository have been used to evaluate the proposed technique. The results of the experiments show that the proposed method is able to determine an optimized set of *k* values for analogy-based prediction, and to give estimates that outperformed traditional models based on a fixed *k* value. The implication is significant in that the analogy-based model will be optimized according the dataset being used, without the need to ask an expert to determining a single, fixed *k* value.

Keywords- *Software estimation, Function Point Analysis, Grey Wolf Optimization Algorithm, Random Forest, Personal Software Process.*

I. INTRODUCTION

The estimation is a process to find the most accurate sizing figure for the software project effort, for example, how many months you will need to develop the software, how many resources you will need to finish the project in the required time [1, 4]. And this translated to money at the end. The estimation is important because it gives the project team some confidence about the required effort and time to plan ahead for the project. Moreover, not all software project is time and material contracts, some of them are fixed cost projects and this estimate will be used as a foundation to negotiate the project cost [13, 15].

1.1 Software Project Estimation

In the development of any software it estimation play an important role and it is a most challenging task. If the estimation of the project is not proper then the development of the software also not in proper way and organized [8][9]. Even when all the factors related to the software development are considered during development process but still projects are not estimated accurately. In this estimation process time of improvement is not calculated. When project is underestimated the effects like under scoping and understaffing affects the project most and project does not meet the deadlines and it loses its credibility [11][12]. To overcome the issues of overestimation and underestimation software project estimation approach is used. If the number of resources is more than required resources it enhances the cost of the project and this condition arise the demand of software project estimation. In small project it is not difficult to estimate the project and mainly estimated by expert judgment approach but in the embedded and large scale projects accuracy and precision of result matters most and they need effective estimation approach. The estimation process with good reliability is an issue that was faced in the projects.

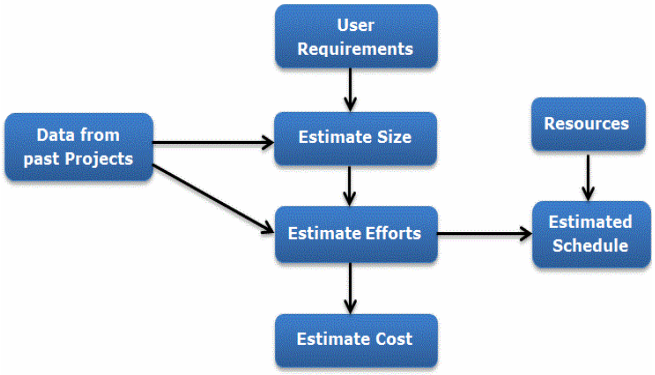


Figure 1: Software Project estimation

In the software estimation process these are the basic steps that are considered:-

- Estimation of project Size: This factor related to the size of th project and measured in the term of function point and line of codes. The UCP (Use case point) and Story points are another method which also helps to estimate the project size [3].
- Effort estimation: Effort estimation [5] for the project based on the manpower and their working hours in the terms of person per month and person hours.
- Scheduling estimation: To decide the total time for project development.
- Cost estimation to decide the overall budget.

1.2 Estimating Size

Effective size estimation is the first step towards a effective product. During the phase of requirement gathering and analysis project size also estimate according to the formal description with client. The cost estimation of the project also depends on the requirement specification and proposal request [2]. The size estimation also depends on the SRS and its details and re-estimation of the size can also be changed according to this in later phases of life cycle.

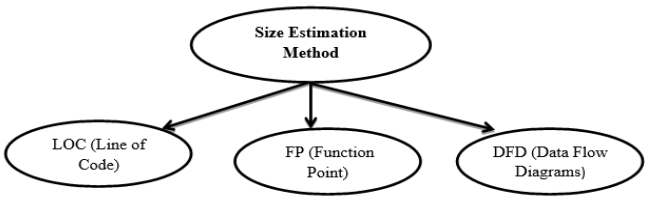


Figure 2: Size Estimation Methods

Following are the two methods that are used for the product size estimation.

(1) Size by Analogy: This method of estimation based on the existing projects and the size estimation. The size of the new project is estimated accordingly because the existing project is similar to new project. This helps to estimate the total cost of project similar to

previous one. BY using the analogy approach only experienced estimator can estimates the better size estimate. This approach work effectively only when we have accurate dimensions of the previous project.

(2) Algorithmic approach to count the product features: The algorithmic approach for size estimation is Function Point which converts the tally into size estimation. This approach based on the classes, modules, function, and methods in the product features.

1.3 Estimating Effort

Effort estimation process starts after the estimation of size of the project. This estimation performed after the complete requirements are defined and size mentioned. The software development process includes the design, develop, and testing of modules and each modules required separate effort to complete it. The coding or development part of software development process takes not more effort than other phases. The writing, documentation, implementation of prototype, and review of document takes more effort [6][7].

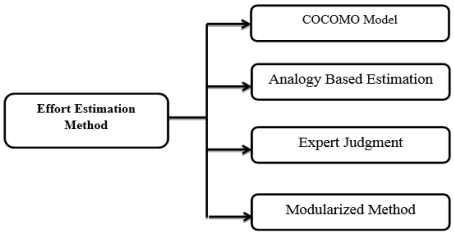


Figure 3: Effort estimation Method

Following are the two methods for estimating the effort from the size.

(1) The existing data of the organization itself is helpful to estimate the project size and costs with respective to each other.

- Documentation of actual results by using existing projects.
- There should be minimum one project in the past which has similar size which helps to determine the estimation of side and then effort.
- The development life cycle of the existing project helps to estimate the development time for new project.

(2) When no similar type of project is available then most accepted and appreciated project. This situation occurs only when no similar project developed earlier. The most commonly used method for effort estimation is COCOCMO and Putnam Methodology. These methods help to converts the size estimation into effort estimation. These models are less effective than the historical project estimation method and their accuracy varies according to the project domain and application areas.

1.4 Estimating Schedule

Schedule of the project describes the working period to complete the assigned task. The schedule estimation done on the basis of total effort calculated for the project. The schedule of the project includes the type of work, starting and ending time. The data gathered from this step used to decide the schedule of the project. In this work is

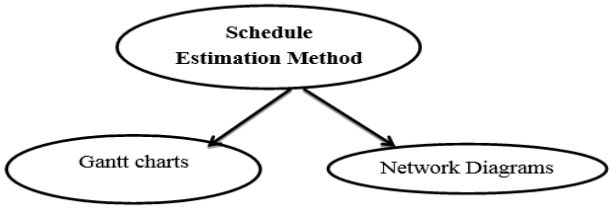


Figure 4: Schedule estimation Method

also broken into modules according to the skill of the persons and timelines for each module is decided.

1.5 Estimating Cost

Cost estimation is process of deciding the budget for the project according to its size and modules. During the cost estimation many factors are considered and the main factors are man power, software on rentals, hardware, office rentals, and telecommunication [2, 10]. The cost estimation depends on the size of the project also because if the project size is large it consumes many resources and manpower and respectively cost is also increased. If the size of project is small it need less resources and less time to complete and its cost is also low. Project cost can be obtained by multiplying the cost of man power per month with estimated effort. After the schedule estimation is completed it is easy to compute the rate per hour for the resources of the project.

II. RELATED WORK

Jodpimai et al. [1] proposed the data mining approach for re-estimation of software efforts. Statistical approach used for the preprocessing of prior phase and selection of input features for learning approach. This model work on four phases that are transformation of data, outlier detection, feature selection, and learning. The result evaluation of the proposed approach done by comparing it with proportion based method and it gives more effective results. Bilgaiyan et al. [2] proposed the genetic algorithm for cost estimation and this method is used to construct the dilation-erosion perceptron to overcome the drawbacks of morphological operators. The performance analysis is done by estimating the 5 different SDCE problem and three metrics. Silhavy et al. [3] Worked on the use case point's estimation by using the subset selection techniques and predict the accuracy of the regression model. Different methods like k-mean, spectral clustering and Gaussian model used for selection of subset. The performance evaluation of the approach done by using two different data sets. The proposed clustering method reduces the prediction error of the regression approach. Benala et al. [4] presented an approach for effort estimation by using the concept of analogy based estimation. The work is based on the differential evolution algorithm and used to optimize the weight of features of similarity functions. The simulation of the proposed work done on promise repository and check the effectiveness of proposed DABE model. This model performs better than PSO, G.A, and neural network. Rao, Ch Prasada, et al. [5] presented the concept of machine learning for effort estimation based on the story points. The effort estimation is based on the functional point, object points and use case points. This approach is applicable on the agile methodology project which increases the chances of the success. The proposed model estimate the effort for the project developed by using agile methodology and machine learning optimize the results for better prediction effort. Dragicevic et al. [6] proposed the Bayesian method for the effort estimation of software development. This model is simple and small and it can be used from the initial stage of the software development. This model is able to estimate the parameters automatically and learned them from the dataset. The data collected from the single company a precision of the model calculated by using different metrics. The statistical results show good prediction accuracy. Moosavi, et al. [7] presented a model which is a combination of bird optimization algorithm and adaptive neuro-fuzzy inference system. Optimization algorithm used to adjust the variables. This model is based on the optimized ANFIS which produced the effective accuracy to estimate the effort on wide range of projects. The test function in this model includes the unimodal and multimodal function. The results evaluation of the proposed work is based on the three models which improves the performance of the model. Masoud, Mohammad, et al. [8] proposed the machine learning algorithm for prediction and estimation. This work is based on the expectation maximization soft clustering method and it is an unsupervised algorithm. This model divides the project into four parts. This project helps to develop enterprise and helps in decision making. COCOMO model is used to test and deploy the model and it provides effective results in effort estimation. Kumar chandan et al. [9] worked on the defect estimation in the software development life cycle. This model based on the Bayesian Belief network which predicts the defect of the requirement analysis, development, coding, and testing. The model developed with the help of expert assessment and qualitative value of software metrics. The model was tested on the 10 project by using qualitative data set. The results of the proposed model were more effective than the existing approach. Md.Akram, et.al [10] detailed an overview of existing software cost estimation techniques or models are given by this model [45]. The models are majorly classified in two type's algorithmic and non-algorithmic models. Key factor in the development of new software is the selection of the suitable cost estimation model and it also depicts the strengths and weakness of various cost estimation models. The main objective is to provide a comparative literature analysis of various cost estimation methods or techniques in this paper. Nassif, et al. [11] presented an approach for the software project estimation by using the linear regression and multi-layer perceptron. The model calculates the software effort on the basis of use case diagrams. The productivity factor was calibrated by using the concept of fuzzy logic in regression model. The estimation based on the productivity of team and software size. The results of this experiment proved that regression model work effectively on small projects and log-linear regression model on large projects. Ekrem Kocaguneli, et al. [12] In order to characterize the essential content of SEE data, i.e., the least number of features and instances required to capture the information within SEE data. If the essential content is very small, then 1) the contained information must be very brief and 2) the value added of complex learning schemes must be minimal. Method: Our QUICK method computes the euclidean distance between rows (instances) and columns (features) of SEE data, then prunes synonyms (similar features) and outliers (distant instances), then assesses the reduced data by comparing predictions from 1) a simple learner using the reduced data and 2) a state-of-the-art learner (CART) using all data. Performance is measured using hold-out experiments and expressed in terms of mean and median MRE, MAR, PRED(25), MBRE, MIBRE, or MMER. Results: For 18 datasets, QUICK pruned 69 to 96 percent of the training data (median = 89 percent). K = 1 nearest neighbour predictions (in the reduced data) performed as well as CART's predictions (using all data). Conclusion: The essential content of some SEE datasets is very small. Complex estimation methods may be over elaborate for such datasets and can be simplified. The experts offer QUICK as an example of such a simpler SEE method. Martin, Sheppard, et al. [13] focused upon building

algorithmic models of effort, for example COCOMO. These can be calibrated to local environments. The experts described an alternative approach to estimation based upon the use of analogies. The underlying principle was to characterize projects in terms of features (for example, the number of interfaces, the development method or the size of the functional requirements document). The process was automated using a PC-based tool known as ANGEL. The method was validated on nine different industrial datasets (a total of 275 projects) and in all cases analogy outperforms algorithmic models based upon stepwise regression. From this work, the experts argue that estimation by analogy is a viable technique that, at the very least, can be used by project managers to complement current estimation techniques.

III. THE PROPOSED METHOD

3.1 Proposed Methodology

Step 1. Input the effort or cost estimation Data set.
Step 2. Initialize the features by Grey wolf search agent.
Step 3. Calculate the fitness value.
Step 4. Find the features weight.
Step 5. Check the Iter < Iter Max if yes go to next step otherwise go to step 4.
Step 6. Update the weight of the features.
Step 7. Initialize the tree after labeling.
Step 8. Select by Bagging and Boosting and make the model for the classification.
Step 9. Analysis the accuracy, precision and recall.

3.2 Proposed methodology: Flowchart

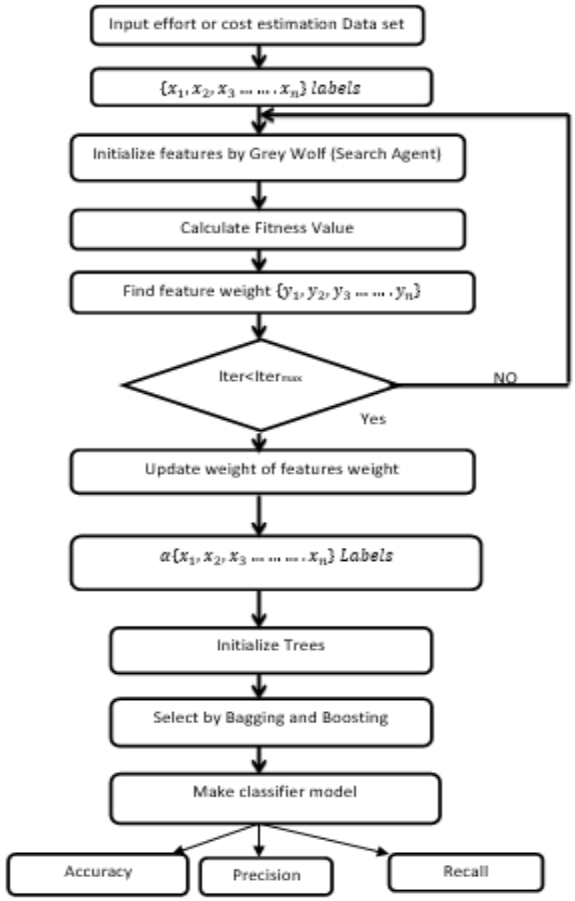


Figure 4: Proposed Flowchart

3.3 Algorithm Used

1. Grey Wolf Optimization Algorithm (GWO): Grey Wolf optimization algorithm is a bio-inspired algorithm which is based on the leadership and hunting behaviour of the wolves in the pack. The grey wolves prefer to live in the pack which is a group of approximate 5-12 wolves. In the pack each member has social dominant and consisting according to four different levels. The below given figure shows the social hierarchy of the wolves which plays and important role in hunting.

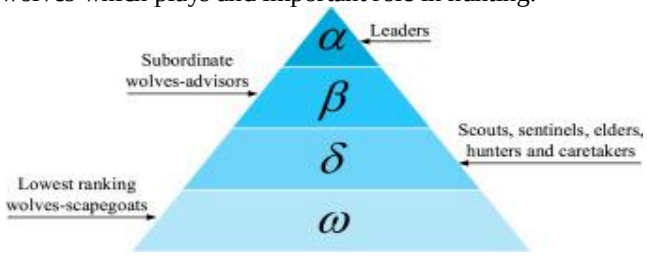


Figure 5: GWO Hierarchy

1. The wolves on the first level are called alpha wolves (α) and they are leaders in the hierarchy. Wolves at this level are the guides to the hunting process in which other wolves seek,

follow and hunt and work as a team. Decision making is the main task that is performed by the alpha wolves and the order by the alpha wolves is followed by all members of the pack.

2. Second level wolves are called beta (β). These wolves are called subordinates and advisors of alpha nodes. The beta wolf council helps in decision making. Beta wolves transmit alpha control to the entire packet and transmit the return to alpha.

3. The wolves of the third level are called Delta wolves (δ) and called scouts. Scout wolves at this level are responsible for monitoring boundaries and territory. The sentinel wolves are responsible for protecting the pack and the guards are responsible for the care of the wounded and injured.

2. Random Forest: Random forest is a learning method for classification, regression and generating the multitude of decision trees.

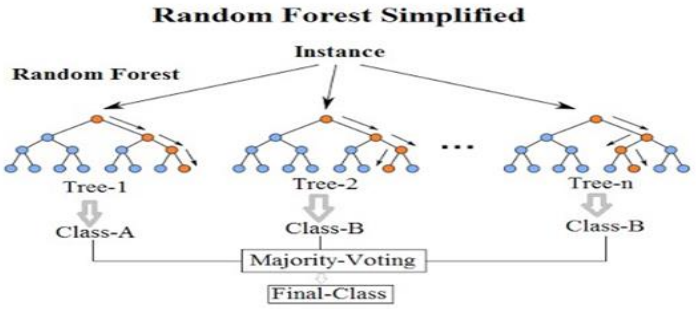


Figure 6: Random forest

It generates the multitude at the time of training and output of the class. It provides the high accuracy and learning is very fast in it. It works very effectively on the large size database. It easily handles the large size input variables without variable deletion.

IV. RESULT ANALYSIS

4.1 Result Analysis

1. Results of Classification

Table I Result of Classification

Classification	Accuracy	Precision	Recall
Random forest + Boost	62	52	69
Random forest + Boost+ GWO	71	93	94
Random forest +Bagging+ GWO	69	68	58
Random forest + Bagging	35	92	97

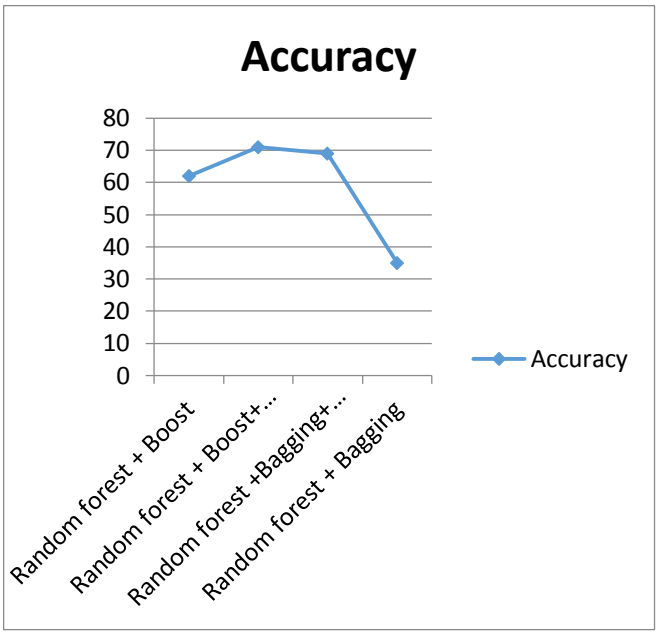


Figure 7 depicts the accuracy of the Random forest + Boost, Random forest + Boost+ GWO, Random forest +Bagging+ GWO and Random forest + Bagging classifiers. The highest accuracy 93 % in graph shown by Random forest + Boost+ GWO and minimum by Random forest + Bagging classifier that is 52%.

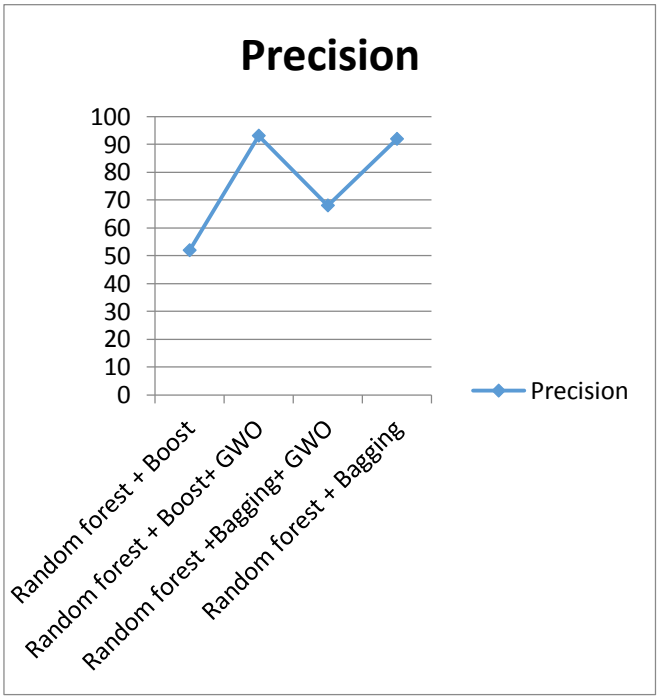
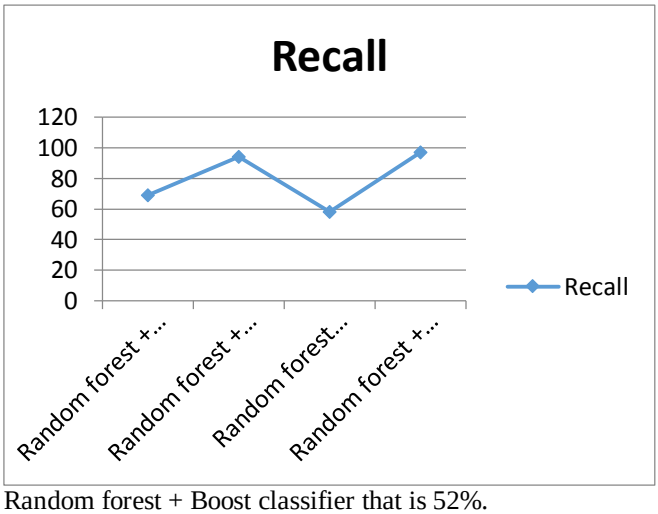


Figure 8: Precision of classifiers

Figure 8 depicts the precision of the Random forest + Boost, Random forest + Boost+ GWO, Random forest +Bagging+ GWO and Random forest + Bagging classifiers. The high precision 94 % in graph shown by Random forest + Boost+ GWO, Random forest + Bagging classifier and minimum by



Random forest + Boost classifier that is 52%.

Figure 9: Recall of classifiers

Figure 9 depicts the recall of the Random forest + Boost, Random forest + Boost+ GWO, Random forest +Bagging+ GWO and Random forest + Bagging classifiers. The high recall 97 % in graph shown by Random forest + Boost+ GWO, Random forest + Bagging classifier and minimum by Random forest + Bagging+ GWO classifier that is 58%.

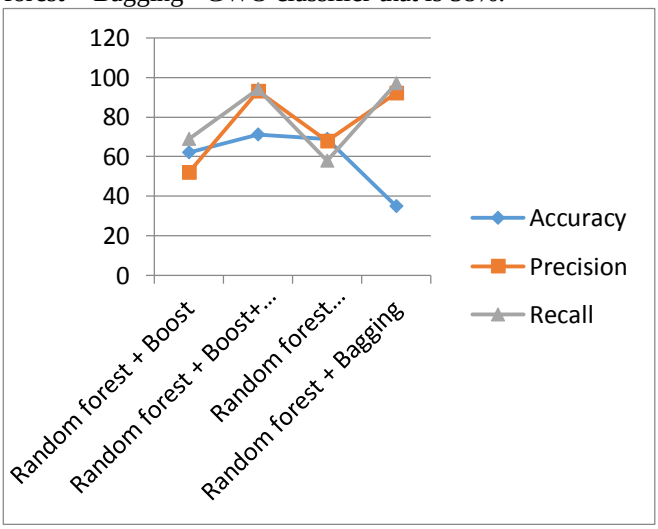


Figure 10Comparison of classifiers

Figure 10 depicts the comparison of the Random forest + Boost, Random forest + Boost+ GWO, Random forest +Bagging+ GWO and Random forest + Bagging classifiers. The effective result shown by Random forest + Boost+ GWO classifier. The red blue curve in the graph represents the accuracy of the different classifiers, Red curve in the graph represents the precision, and green curve represents the recall of the classifier.

2. Random Forest Regression

Table II Random Forest Regression

Random Forest Regression	Accuracy
RF+ GWO	79
RF	52.10

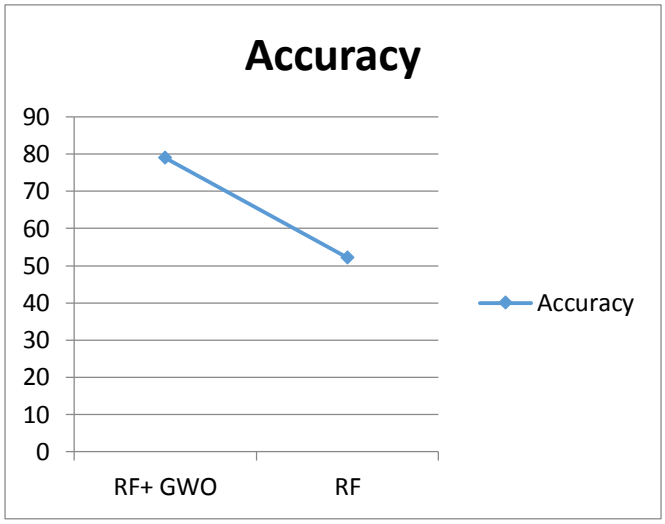


Figure 11: Accuracy of the classifier

In figure 11 accuracy comparison is shown with Random forest and Random forest with GWO. The x axis of graph represents the classifiers and y axis of graph represents the random values of accuracy. The accuracy of the Random forest with GWO is better than random forest.

3 Result Screenshots

Random Forest

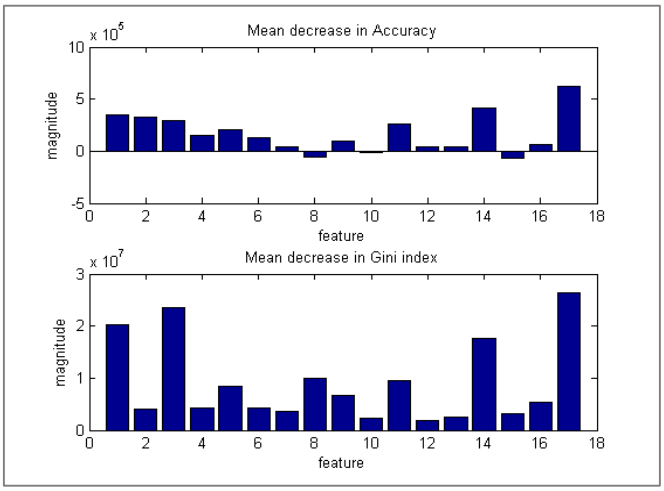


Figure 12: Mean Decreases in Accuracy

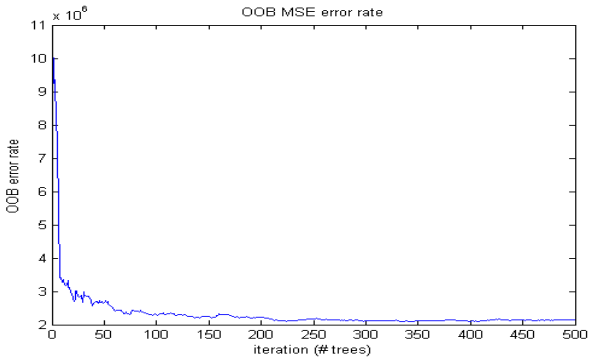


Figure 13: OOB MSE error rate

Random Forest+ GWO

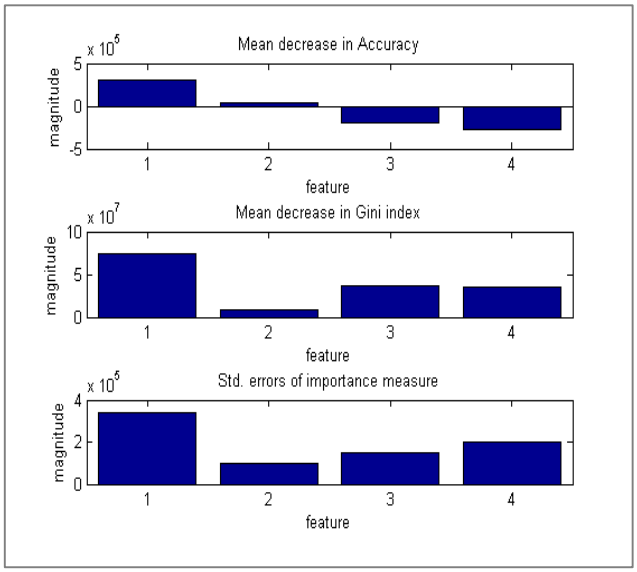


Figure 14: Mean Decreases in Accuracy

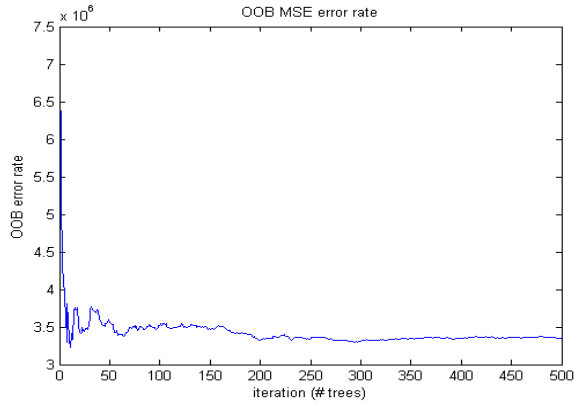


Figure 15: OOB MSE error rate

IV CONCLUSION

To conclude, this study proposed a novel approach by applying hierarchical clustering to determine the best value of k for analogy based software effort estimation (ABE) before its model building, largely removes the need to arbitrarily define a fixed k in a traditional analogy based setting. Result shows that the proposed technique utilizing hierarchical clustering effectively determine the most suitable value of k for the testing case i, which would result in a more optimized ABE model for software effort estimation. The proposed method is well defined and fully automated that is able to determine the value of k for ABE automatically, and thus a major improvement to analogy based software effort estimation.

V REFERENCES

[1] Jodpimai, Pichai, Peraphon Sophatsathit, and Chidchanok Lursinsap. "Re-estimating software effort using prior phase efforts and data mining techniques." *Innovations in Systems and Software Engineering* (2018): 1-20.

[2] Bilgaiyan, Saurabh, et al. "Chaos-based Modified Morphological Genetic Algorithm for Software Development Cost Estimation." *Progress in Computing, Analytics and Networking*. Springer, Singapore, 2018. 31-40.

[3] Silhavy, Radek, Petr Silhavy, and Zdenka Prokopová. "Evaluating subset selection methods for use case points estimation." *Information and Software Technology* 97 (2018): 1-9.

[4] Benala, Tirimula Rao, and Rajib Mall. "DABE: Differential evolution in analogy-based software development effort estimation." *Swarm and Evolutionary Computation* 38 (2018): 158-172.

[5] Rao, Ch Prasada, et al. "An Agile Effort Estimation Based on Story Points Using Machine Learning Techniques." *Proceedings of the Second International Conference on Computational Intelligence and Informatics*. Springer, Singapore, 2018.

[6] Dragicevic, Srdjana, Stipe Celar, and Mili Turic. "Bayesian network model for task effort estimation in agile software development." *Journal of Systems and Software* 127 (2017): 109-119.

[7] Moosavi, Seyyed Hamid Samareh, and Vahid Khatibi Bardsiri. "Satin bowerbird optimizer: A new optimization algorithm to optimize ANFIS for software development effort estimation." *Engineering Applications of Artificial Intelligence* 60 (2017): 1-15.

[8] Masoud, Mohammad, et al. "Software Project Management: Resources Prediction and Estimation Utilizing Unsupervised Machine Learning Algorithm." *International Conference on Engineering, Project, and Product Management*. Springer, Cham, 2017.

[9] Kumar, Chandan, and Dilip Kumar Yadav. "Software defects estimation using metrics of early phases of software development life cycle." *International Journal of System Assurance Engineering and Management* 8.4 (2017): 2109-2117.

[10] Akram, Md, Reddy G, Devi Prasad, & Sriramaneni, Ravi Teja. "Literature Survey on Various Software Cost Estimation

Models". *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, Volume 4, Issue IV, 2016.
[11] Nassif, Ali Bou, Danny Ho, and Luiz Fernando Capretz. "Towards an early software estimation using log-linear regression and a multilayer perceptron model." *Journal of Systems and Software* 86.1 (2013): 144-160.
[12] Kocaguneli, Ekrem, Tim Menzies, Jacky Keung, David Cok, and Ray Madachy. "Active learning and effort estimation: Finding the essential content of software effort estimation data." *IEEE Transactions on software engineering* 39, no. 8 (2012): 1040-1053.
[13] Shepperd, Martin, and Chris Schofield. "Estimating software project effort using analogies." *IEEE Transactions on software engineering* 23, no. 11 (1997): 736-743.
[14] Kemerer, Chris F. "An empirical validation of software cost estimation models." *Communications of the ACM* 30, no. 5 (1987): 416-429.
[15] Software Project Estimation. [Online]. Available at: https://courses.cs.washington.edu/courses/cse403/07sp/assignments/estimationbasics.pdf [Accessed on 14-06-2019].