Machine-Learning Based Model for Improving Effort Estimation using Risk

Ramakrishnan N, Girijamma H. A., Balachandran K.

Abstract: In the real world, many software projects have effort, schedule, and cost overrun. Estimation accuracy is vital for a successful software project. This is a challenge since mostly there is a tendency to over-estimate or under-estimate the size and effort needed in a software project. The prediction of required effort is a critical activity, and a greater focus is required by considering additional factors such as project risk. The importance of project risks and their management has been indicated in literature and its consideration in effort estimation is the focus of this research. This research paper has a focus on proposing a model based on machine learning techniques for improving the effort estimation through inclusion of risk score. The proposed solution considered aggregating the capability of various machine learning models for prediction. The methodology involved usage of extreme gradient boosting algorithm. Data for the research included projects from industry standard dataset and organizational projects. The analysis revealed a reduction in the root mean square error values over multiple iterations suggesting an improvement in model performance. This reveals a better estimation due to minimization of the gap between estimated and actual efforts in a project. This, in turn, would enhance the potential for success of the project through an improved estimation process integrating risk score along with other parameters for estimation of software development effort. The feature importance chart also revealed that project risk score is an important attribute to be considered for effort estimation.

Keywords: Estimation, estimated effort, actual effort, project risk, machine learning, predictive model

I. INTRODUCTION

The estimation of required effort plays a significant role in the entirety of SDLC (Software development life cycle). Prediction of effort estimates helps to have proper project planning and control. A software estimate could be indicated as a range rather than a single number to have more accuracy. Measurement of risks is required to find out an estimation range for schedule and cost. The base cost which is one of the components of estimates does not include risks or contingencies[1]

Revised Manuscript Received on February 28, 2020.
***** Correspondence Author

Ramakrishnan N.*, Research Scholar and Associate Professor, Institute of Management, CHRIST (Deemed to be University), Bangalore, India.
Dr. Girijamma H. A., Professor, Department of Computer Science and Engineering, RNS Institute of Technology, Bangalore, India.
Dr. Balachandran K., Professor and Head, Department of Computer Science and Engineering, CHRIST (Deemed to be University), Bangalore, India.

There are many instances where software projects do not deliver acceptable systems within the estimated cost and schedule. A majority of these failures could have been averted by proper assessment and mitigation of risks. Software risk exists because of uncertainty and due to inherent complexity of the development scenario.

Risks are of two types namely many known and unknown things that cannot be incorporated in the project plan which could result in increased effort / cost and further leading to poor quality software. Risk Management corresponds to the complexity and size of the project that is being considered. When compared with cost estimation approaches based on risks that apply a factor or unknowns, identification and quantification of specific event risks are performed[1].

The key elements of risk management include identification of risk factors, assessment of probabilities of risks and their impact on the project, development of risk mitigation strategies to manage the identified risks, monitoring of risk factors, and establishing a contingency plan.

II. THEORETICAL BACKGROUND

The estimation of effort for developing software is a vital activity, but it has not been adequately mastered[2]. Better estimates enable Project managers to plan appropriately for a software project [3]. Both under-estimation and over-estimation of effort could impact the success of a project. Uncertainty in a software project is required to be accounted to have improved estimation accuracy. [4].

Estimation of software development effort is performed for various purposes. An accurate effort estimate facilitates effective monitoring and efficient scheduling of tasks and resources [5].

Various approaches for estimation of effort have been established and followed in the past. They could be categorized into four main aspects (a) Methods based on expert judgment such as Delphi rely on intuition of experts and experience obtained from projects executed in the pas (b) Methods based on analogy that involve identification of historical projects that have similarity to the project under consideration, and utilizing the known effort values to arrive at the estimated effort . (c) Methods based on Parametric modelling such as COCOMO that depend primarily on equations that have been elaborated with the help of historical data and indicating the effort as a function of discriminant parameters that influence the effort, and known as effort drivers. (d) Methods based on machine learning and utilizing techniques pertaining to artificial intelligence such as Fuzzy Logic and Neural Networks (NN) that establish a model of the complex relationship between effort drivers and the effort[6].

Many parametric methods for estimation of effort, such as COCOMO, have their limitations and challenges.

As they are based on historical data and pertaining to specific organizations, difficulty arises in adapting these to new project situations and environments. Further, there exists an inability to model the complex relationships existing between effort and effort drivers in an effective manner. Methods used for effort estimation that are based on techniques related to artificial intelligence, such as Neural Networks, are able to handle a majority of such problems. Models for effort estimation based on Neural networks have the ability of learning from past data, being adapted to suit any organizational and project scenario, to be updated over a period of time and for modelling complex relationships[6].

It has been perceived by practitioners that there is a link between risk management practices and value creation in software projects [7]. Risk is neither conceived adequately nor performed to the required levels in practice [8].

The process of developing software involves gradual refinement and making effort estimates in the early phases of a project's lifecycle. One of the mistakes (sins) in software estimation is "not including risk impacts in estimates"[9]. The quantum of risks corresponding to each activity in a software development project substantially influences the success of the project. Risks exist in all software projects, and it is highly essential that stakeholders ensure identification, understanding, and mitigation of risks that might hamper the success of the project. If risk mitigation is performed as a centralized activity, there are greater chances of success of the efforts for software development. The identification and aggregation of risks is a key predictive method to capture the probability of unplanned or inadmissible events experienced in a software project[10].

A study on effort estimation and risk analysis reveals that "risk and effort estimation go hand in hand" and it is important to be considered in the estimation process[11]. In a study on software development effort estimation it is stated that risk is also an important parameter that contributes to reasons for deviations between effort estimates and actual effort[12]. It is indicated in many studies that estimates done at the start of the project will be less accurate than the estimates done at the later phases of the project, due to involvement of risks in the various stages. Estimation acts as a framework for controlling the project through improved planning and scheduling.

III. RESEARCH METHODOLOGY

The research approach involved consideration of a dataset with 115 Projects from the NASA dataset and from organizational projects data set. The NASA dataset is an industry dataset and made available for research purposes. In addition, data from organizational projects were included to introduce diversity and minimize any bias. The attributes in the dataset included Project Size (in Kilo Source Lines of Code), Project Risk score, Staff application knowledge, Software Logical Complexity, Customer Participation, Staff team skills, Usage of tools, Estimated effort, Contingency Allowance percentage. All these attributes constituted the independent variables, with Actual effort being the dependent variable. Data pre-processing was done to eliminate those records having missing values for attributes in the dataset. Data validation was done on the dataset to identify existence of any anomalies / missing information. The dataset was subjected to statistical tests for multi-collinearity and autocorrelation.

A predictive model requires creating a split of the existing dataset into training and test datasets. The dataset was split such that 80% of the data constituted the training dataset and the balance 20% constituted the test dataset. As part of predictive modeling, the approach was to arrive at a solution based on aggregation of the predictive capability of many machine learning models. Techniques such as Gradient boosting which are ensembles of decision tree methods provide estimates pertaining to importance of features by default from a trained predictive model[13].The boosted trees were constructed and the importance for each of the attributes were constructed. It indicates the usefulness or value of every feature while constructing the boosted decision trees within the model. Larger the extent to which an attribute is utilized for making key decisions with reference to decision trees, greater would be its relative significance. The importance is computed for every attribute existing in the dataset. This allows attributes to be ranked and compared with each other[13].

An upgraded version of gradient boosting known as Extreme Gradient Boost algorithm was utilized for machine learning. A factor 'f' exists for every node and t(x) was multiplied to this factor. Fitment of each $t_i(x)$ was done to the gradient obtained in every step. The boosted function $F_i(x)$ for each iteration was obtained by deriving the multiplying factor 'f_n' for each terminal node. This could be represented as shown below:

```
Begin
    For i = 1 to n    //Comment: Iteration of the steps
        **Step 1**:
            Multiply Factor ($f_i$) for the terminal node with the
            additive model ($t_i(x)$)  //Comment: Step for boosting
            of the decision tree through learning from residuals
            of the current model
        **Step 2:**
            Boosted Function $F_i(x)$ = Function in previous step
            ($F_i$-1(x)) + multiplied value $f_i t_i(x)$ //Comment:
            Construction of the boosted tree and generation of
            the updated model
        **Step 3:**
            Computation of RMSE-mean  //Comment: Root
            mean square error
            If RMSE-mean value is lower than in previous
              iteration  //Comment: Improved model
              performance
            Then
              Go to Step 1 //Comment: Repeat the
              iteration steps
    Next
End    //Comment: No decipherable pattern in residuals
indicating a random distribution, and hence end the iteration
and exit
```

The detailed process is described below:

Based on the initial model M_0, an updated model M_1 was generated.

The new model M_1 was fit to the residuals from the function $F_0(x)$ applicable for the initial model M_0. An additive model $t_1(x)$ was created as a regression tree using the residuals from the function $F_0(x)$. This helped in the reduction of residuals from the previous step. The boosted function $F_1(x)$ was arrived at by aggregating / combining with the additive model $t_1(x)$ which had a learning from the residuals of the function $F_0(x)$ as shown below:

$$F_1(x) <- F_0(x) + f_1t_1(x)$$

The additive model $t_1(x)$ had a learning from the residuals of $F_0(x)$. A combination of the initial model M_0 and the additive model t_1 yielded M_1 constituting the boosted version or second stage of the model M_0. It was observed that the mean square error (MSE) in the boosted model M_1 was lesser than the MSE in the base model M_0. With every iteration, the MSE of the model was found to be lesser when compared to the earlier version of the model (i.e., the mean square error from model M_2 was lesser than that of M_1 and MSE of M_3 was lower than that of M_2. In each iteration, training of the learner happened based on the residuals in the previous iteration. M_n is the stage of the model after the n^{th} iteration and is a boosted version of the model M_{n-1}. The mean square error from M_n was found to be lesser than that in M_{n-1}. This indicates that the performance of the model had improved while comparing with the previous iterations. A scenario occurs wherein after a series of iterations, the residuals seem to have no decipherable pattern, and a random distribution is observed. At this stage, iteration could be ended considering achievement of the maximum possible accuracy.

IV. ANALYSIS, RESULT AND DISCUSSION

Anderson-Darling test was performed and, the data was found to have a non-normal distribution. Based on statistical tests, it was established that multi-collinearity and autocorrelation do not exist. Spearman's rank correlation test was performed on the dataset and the output in terms of Spearman's rho (correlation coefficient) revealed the following:

The extent of correlation of the various attributes on project risk score ordered from highest to lowest are Contingency Allowance percentage, Estimated effort, Size in Kilo source lines of code, Staff team skills, Actual effort, Staff application knowledge, Usage of tools, Software Logical Complexity, and Customer participation.

Project risk has a correlation coefficient of 0.160 with Project size (in Kilo Source Lines of Code) indicating that while there is a positive correlation, it is quite low. This indicates that the size of a project does not have a significant impact on the project risk score. In other words, bigger projects need not necessarily have a high project risk score and vice versa. It is observed that the correlation coefficient between project risk and estimated effort is slightly higher than the correlation coefficient of project risk with most of the other variables.

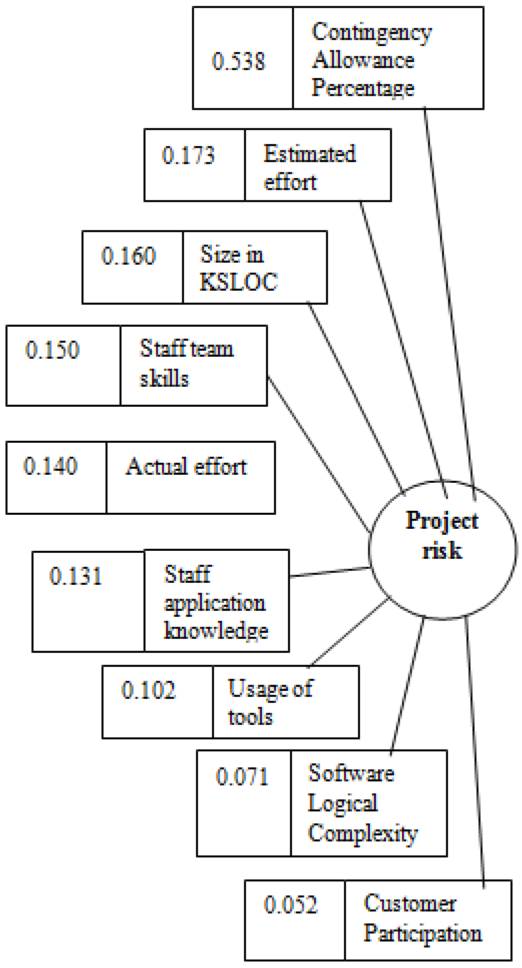

Fig. 1. Correlation of Project risk with other variables

As part of predictive modeling, the approach was to arrive at a solution based on aggregation of the predictive capability of many machine learning models. Extreme Gradient Boosting was used for this purpose. Techniques such as Gradient boosting which are ensembles of decision tree methods automatically provide estimates of feature importance from a trained predictive model. The boosted trees were constructed and the importance for each of the attributes was constructed. It indicates how useful or valuable each feature is in the construction of the boosted decision trees within the model. The more an attribute is used to make key decisions with decision trees, higher is its relative importance. The importance is calculated explicitly for each attribute in the dataset allowing attributes to be ranked and compared with each other.

Table I: Feature importance (F-score values)

Estimation parameter (feature)	F-Score
Project size (in Kilo Source Lines of Code)	33
Contingency allowance percentage	18
Project risk score	15
Staff application knowledge (project team members' knowledge of the application)	12

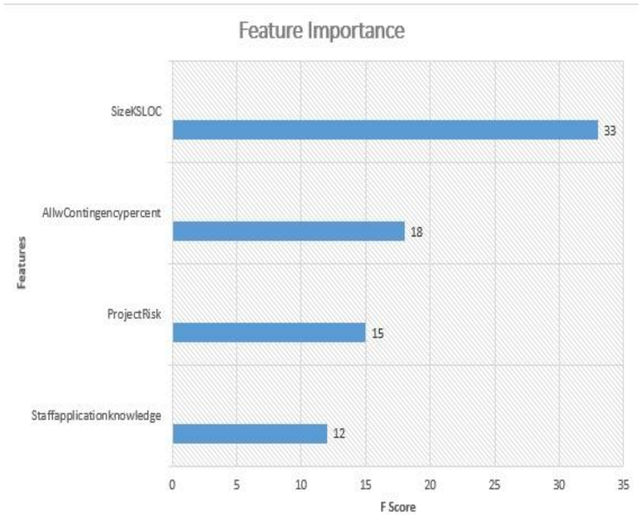

Fig. 2. Feature importance chart

A 'feature importance' chart is utilized to visually depict the relative importance of every feature in the dataset. The X-axis indicates the F-score and the Y-axis has the important attributes used in effort estimation. The feature importance chart revealed that Project size (in Kilo Source Lines of Code) has the highest relative importance, and this is understandable. The relative importance of ProjectRisk score is greater than the value for Staff application knowledge. This reveals that when Staff application knowledge is considered in computation of effort estimates, Project risk score which has a higher relative importance (higher F Score) is not being considered. This validates the need to consider Project risks in the effort estimation process, and the inclusion of Project risk score in the computation of effort estimates.

As part of the analysis, the model was subjected to three fold cross-validation. The observed RMSE-mean values and the reduction in values from the initial iteration are as depicted in the following table:

Table 2: RMSE-mean and reduction from initial value

Iteration	RMSE-mean	Reduction in RMSE-mean value (from initial iteration)	% reduction in RMSE-mean value (from initial iteration)
Iteration 1	1088.45		
Iteration 2	1059.30	28.65	2.63%
Iteration 3	1033.07	55.38	5.23%
Iteration 4	1005.87	82.58	7.99%
Iteration 5	986.75	101.7	10.11%
…..	……	---	----
Iteration 49	650	438.45	40.28%

The reduction in RMSE values in the boosted model with each iteration indicates an improved model performance that would minimize the gap between the values for estimated effort and actual effort.

V. CONCLUSION

There are various issues associated with effort estimates in a software project. The need for enhancing the effort estimation process to arrive at better estimates was emphasized through literature studies. Though various estimation techniques and approaches exist, the deviation between estimated effort and actual effort occurs to a large extent. This was also evident in the industry standard dataset and organizational dataset used for the research. The importance of effort estimation and the process of managing risks in a software project provided the focus for this research by means of a risk-based estimation approach. Correlation tests revealed that project size does not have a significant influence on the risk score of the project. Extreme gradient boosting algorithm as a part of machine learning was used for predictive analysis on the dataset with a risk-integrated effort estimation approach. The integration of risks into effort estimation model resulted in decreasing RMSE-mean values with each iteration and this revealed improved model performance. The feature importance chart also indicated that risk score is also an important factor for effort estimation. It could be concluded from the results of analysis that incorporation of risks while estimating effort for a software project would lead to an enhanced effort estimation approach. It could also potentially reduce the gap between estimated and actual efforts in a software project.

REFERENCES

1. El Koutbi, S., Idri, A. "Software Efforts Estimation Risk Management over Projects Portfolio", Computer and Information Science, vol.11,Issue 4,2018.
2. Hari V.M.K., Tegjyot.S.S., Kaushal, B.S.S., and Jagadeesh. M, "SEEPC: A Toolbox for Software Effort Estimation using Soft Computing Techniques", International Journal of Computer Applications , vol.31,Issue 4,2011,:pp12-19.
3. Abdelali Z., Mustapha, H., and Abdelwahed, N.. "Effort Estimation Using Random Forests: An Empirical Study and Evaluatiion", International Journal of Intelligent Engineering and Systems, Vo;11,Issue 6, 2018,pp.. 300-311.
4. Boehm B, Abts C, and Chulani S, "Software development cost estimation approaches – A survey",. (2000).,Annals of Software Engineering, vol. 10,pp.177-205.
5. Jorgensen, M., "A review of studies on exper estimation of software development effort", The Journal of Systems and Software. 2004,pp.37–60
6. 6.Shepperd, M., Schofield, C., and Kitchenham, B. ICSE '96 Proceedings of the 18th international conference on Software engineering, 1996, ISBN:0-8186-7246-3,pp. 170-178.
7. Srinivasan, K., Fisher,D..,"Machine Learning approaches to estimating software development effort", IEEE Transactions on Software Engineering, vol. Issue 2,1995,pp. 126-137
8. Ahmed, M.A., Muzaffar, Z.,. "Handling imprecision and uncertainty in software development effort prediction: A type-2 fuzzy logic based framework", Information and Software Technology, vol.51,Issue 3,2009, pp.640-654.
9. Park, H., Back, S.,"An empirical validation of a neural network model for software effort estimation,",Expert Systems with Applications, 2008,vol.35,Issue 3,pp.929-937
10. Idri,A., Zakranil, A,,and Zahi,A,,"Design of Radial Basis Function Neural Networks for Software Effort Estimation" , International Journal of Computer Science Issues, 2010,vol.7, issue 4,,pp.11-17,
11. Setiono R, Dejaeger, K, Verbeke W., Martens, and D, Baesens, B.. "Software Effort Prediction Using Regression Rule Extraction from Neural Networks", 22nd International Conference on Tools with Artificial Intelligence, 2010, vol. 2, 45-52

12. Willumsen, P., Oehmen, J., Stingl, V.,and Geraldi, J., Value creation through Project Risk management, *International Journal of Project Management*, 2019. https://doi.org/10.1016/j.ijproman.2019.01.007
13. Bannerman, P.L., “Risk and Risk management in software projects: A reassessment”, *The Journal of Systems and Software*, 2008, pp.2118-2133.
14. Steve McConnell,“10 Deadly Sins of Software Estimation” Retrieved from http://www.construx.com on 23 Dec 2019
15. Patil M.V. , “Software effort estimation and risk analysis - a case study”,“*IET-UK International* Conference on Information and Communication Technology in Electrical Sciences *(ICTES 2007)*” 2007.
16. Jorgensen, M., “A review of studies on expert estimation of software development effort “, *The Journal of Systems and Software*,2004,,pp. 37-60
17. Basten, D.,Mellis, W..,“A current assessment of software development effort estimation”:, *International symposium on empirical software engineering and measurement*, 2011,pp.235–244
18. El-Masri, M., Rivard, S.”Specifying the Software Project Risk Construct”, *AMCIS 2010 Proceedings*”,2010, p.468.

AUTHORS PROFILE

Ramakrishnan N., is presently a Research Scholar and Associate Professor at Institute of Management, CHRIST (Deemed to be University), Bangalore. He has more than 30 years of experience, out of which around 23 years has been in the IT industry in various roles. He has an M.S. in Software Systems from Birla Institute of Technology and Science, Pilani. His areas of interest include Information Systems, Software Engineering, and Machine Learning.

Dr. Girijamma H. A.., is currently working as a Professor in the Department of Computer Science and Engineering at RNS Institute of Technology, Bangalore. She has 26 years of experience in teaching. Her research interests are in the areas of automata, compilers, fuzzy logic, data analytics, software engineering, image processing , natural language processing and Machine learning.

Dr. Balachandran K., is a Professor and Head of the Department of Computer Science and Engineering, CHRIST (Deemed to be University),Bangalore. He has a PhD from Anna University, and his areas of interest include data mining and related algorithms. He has published many research articles in journals and presented papers in International and National Conferences.