International Journal of Computational Intelligence Research
ISSN 0973-1873 Volume 19, Number 1 (2023), pp. 13-23

https://dx.doi.org/10.37622/IJCIR/19.1.2023.13-23

Future of Project Management with Machine Learning

[1]R. Mamatha, [2]Dr.Lalitha Surya Kumari, [3]Dr.A.Sharada*

1 Research Scholar, 2 Professor, Computer Science & Engineering,
KoneruLakshmaiahEduaction Foundation,
Deemed to be University, Hyderabad,Telangana, India.
Email. id: [1]mamatha.racharla@gmail.com
Email. id: [2]vlalithanagesh@gmail.com
Email. id: [3]sharada@gnits.ac.in

ABSTRACT

Software companies will constantly work on numerous projects at a time. Project managers are responsible for making each project successful. As to monitor and control many projects organizations use project management tools to perform tasks in a well-organized manner. Current day's lot of data is generated in organizations related to software development as the organizations use project management tools to track the status of the project. This project data generated by project management tools like Jira, Asana, and Trello can be used to solve resource allocation problems. Though there are numerous solutions for addressing the project scheduling problem, none of them has intelligent decision-making. Research is still scarce in the area of software project management and evidence is required to assess the theoretical methods and ideas. This paper introduces a resource allocation method that uses machine learning for effective project scheduling.

Keywords: Project management, Machine learning, Productivity.

Introduction

There are various aspects of project development. In the IT industry, each aspect has different functionality and each of which has different features. This work will mainly focus on the resource management aspect which plays a crucial role in project development. Any Project life cycle consists of the Engineering phase and Management phase. There are many project management tools available for managing and controlling the team.

Reasons for using project management tools:

1. To plan easily: A project management tool will provide the team to easily establish a hierarchy of tasks for efficient completion. It also indicates the relation between the tasks so that the team leader decides on which tasks to assign to whom.
2. Managing tasks efficiently: Project management tools are very useful to assign tasks to the team and monitor their performance with better task management it is ensured that the team is working in unity.
3. Continuous team workflow: Some big projects comprise more people, where the team leader includes several people to work collectively, and creating cohesion among team members is a challenging task. Through project management tools this can be done seamlessly.
4. Sharing information, and calendars: Project management tools support storage and maintaining a variety of information regarding the project, if the team has a centralized document storage plan all the team members can easily access the data. As well having a team calendar makes coordination easy this ensures the remote team communicates with other team members without any hassle.
5. Accurate project tracking: project management tool allows team leaders to collect data about their team and know their work pace. This makes planning much easier and also results in better performance.

Apart from the advantages, there are a few disadvantages of using project management tools:

All of the project management tools available will cost money to install and maintain, and some programs are sold with additional modules that can be costly. For the project manager, learning the tool and ensuring that their team uses it smoothly may take some time. Some of the most recent versions of the utility allow multiple user access, which could lead to illegal access to sensitive information. Some software management tools also have useless capabilities that firms don't utilize but for which they have to pay a lot of money. As a result, a model that uses a machine learning method to manage the job allocation problem using the project's historical data is required.

An ML-enabled system that is used to manage day-to-day management tasks without any need for human input. This will automate mundane tasks and develop an understanding of project performance that is used to find insights into projects and make smarter decisions. In the current scenario, IT companies are waiting for ML-enabled project management software. None of the present tools have ML capability. It is high time to address this problem using machine learning as a solution.

Automating project management has the potential advantage of increasing scalability, improving risk assessment capabilities, and more effective communication. Any project that can automate management activity will have a scope of relieving from doing the repetitive task and take the opportunity to give greater output to the organization. There are projects in a company that needs continuous monitoring and managing. If Machine learning is used to make smarter decisions, then managers and

team leads will get a break from doing repetitive tasks and focus on improving the productivity of the project.

RELATED WORK

2.1 Research on Machine Learning Techniques in Project Management:

This section will focus on the previous work done on the projects using Machine learning methods. Some ML algorithms like KNN, ANN, and Fuzzy logic were used to estimate the effort, and cost related to real projects.

Idri, et al. [1] employed the K-nearest Neighbour Algorithm (KNN) to study the handling of missing values (MV) in software engineering data systems (SE) The adoption of Machine Learning (ML) techniques in coping with missing values in software engineering data has increased, according to the study. This study showed the distribution of ML and non-ML techniques in software engineering over the years which can be seen in Table 2.1 below.

Table 2.1: Project done using machine learning.

Score	No. of Projects
Aim clearly defined	< 1000 projects
Aim partially defined	Ranging from 100 to 1000
Aim not defined	< 100

The majority of the research was historical in nature, with the International Software Benchmarking Standard Group (ISBSG) data being the most widely used.

Machine learning techniques used to anticipate requirement volatility have been summarized by Alsalemi and Yeoh [2]. Fuzzification, Bayesian networks, Binary classification, and Artificial neural networks were among the machine learning approaches used by the researchers. The majority of software development is done in iterative models, and every project has needs that change frequently. According to the findings, project failure is caused by changes in requirements. Machine learning methods such as ANN and logistic regression use independent variables retrieved from the requirement document and perform a regression analysis to determine the estimation of independent variable correlation. Different independent variables, such as the size of requirements, the size of requirements documents, use cases for technical mapping, and complexity, will act as independent variables, and the most relevant attributes for requirement volatility are type of module, dependency among requirements, and type of requirements.

Pillai, et al. [3] When utilizing the academic dataset, linear regression has the most accuracy in terms of estimating method, but when using global datasets, it has the lowest accuracy. Agile initiatives are getting more accuracy by depending on internal datasets, according to research. Small businesses profit from depending on a limited collection of internal project data. Due to the inefficiency of representing environmental factors, the regression-based estimate is the most common method.

Function points (FP) are best employed for size estimation rather than effort estimation.

Sharma and Singh Due to their learning nature, machine learning models have been shown to give excellent accuracy. Machine learning algorithms such as Random Forest (RF), Multi-Layer Perceptron (MLP), and Support Vector Machine (SVM) are utilised in this work to estimate project effort. The technical variables affecting the projects are observed, and weights corresponding to elements are assigned, and the environmental complexity factor is derived using eight components stated in Table 2.2. To calculate effort, the Use-case point approach was employed.

Table 2.2: Technical factors and their corresponding weights.

Environmental Factor	Description	Weight
e1	Analyst capability	0.5
e2	Motivation of Team	1
e3	Experience in object-oriented programming	1
e4	Constant requirements	2
e5	application experience	0.5
e6	Part-time workers	-1
e7	Tough programming languages	-1
e8	Familiarity with the objectary	1.5

Later in the future machine learning techniques like RF, SVM, and MLP. The size of the project, Complexity, and Productivity are the input values given as input to all these models. The models were evaluated using different evaluation criteria, results are compared. It is observed that among all three model's RF model performed better than MLP and SVM. On the other side, it is observed that the performance of SVM and MLP were very close and satisfactory when compared to RF. The maximum accuracy obtained by MLP and SVM was 86.046%. It is suggested that the study be expanded in the future using other machine learning models, such as XBoost, and verified with more diverse datasets.

Shivhare and Rath Effort estimation is based on non-quantitative data and machine learning algorithms, according to this research. The Naive Bayes classifier is used to perform the estimation. The performance of these techniques is assessed and compared using three parameters: Mean Magnitude of Relative Error (MMRE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE), and it is discovered that when compared to other Neural Network techniques, Nave Bayes classifiers outperform them all.

Malhotra and Chug A systematic analysis of current software-maintenance research was conducted by Not Mentioned. The study found that the use of machine learning techniques has increased. This paper has given empirical evidence in a more comprehensive form regarding various factors which affect the software maintainability, different methods to improve it, identifying various methods to

improve maintainability, and comparing the performance of different maintainability prediction models.

Srikumar. P,et.al[7] has stated that Analogy based estimation models alone are not so accurate when compared to the Machine learning models. It is observed that when these analogies-based and machine learning models are combined to find the cost estimation led to better results. The use of function point is vanishing to estimate size due to its inability to represent the environmental parameters, since the last few years the trend is towards the use of machine learning algorithms with more accuracy.

Machine learning approaches are employed in the estimation and prediction of the engineering phase, mostly for project effort estimation, according to earlier work. The proposed work suggests the usage of ML in the project management phase where these techniques are used to map the skills of employees with the experience, they have worked in past projects so that their skills are utilized in an efficient way when new tasks enter the team.

Results of Previous Study

Overall results from previous work are shown in Table 2.3. along with the graph in Fig.2.1.

Table 2.3: ML techniques used in software projects.

S.No	Technique used	Results of the study
1	KNN	Management of missing values in the dataset [1]
2	Bayesian network	Change requirements [2]
3	Random forest	Effort estimation [3]
4	Naïve bayes classification	Effort estimation [4]

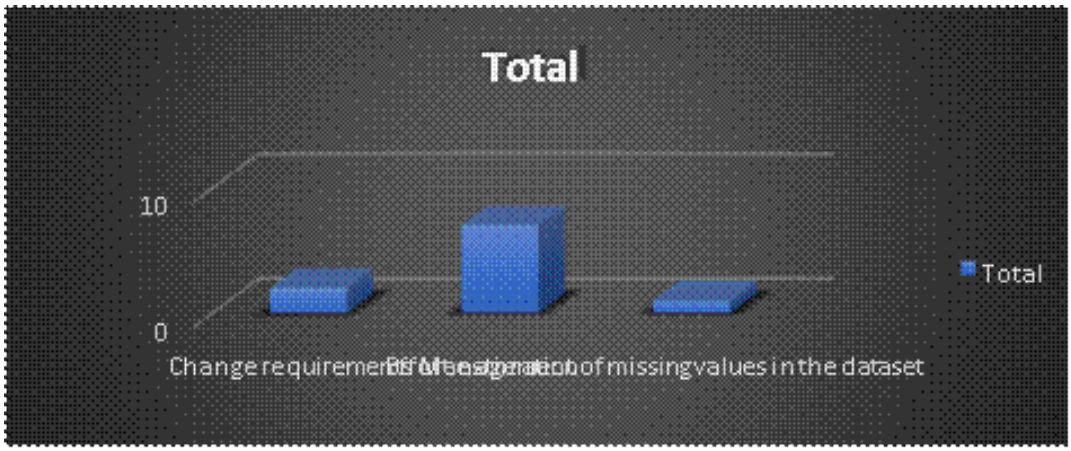

Fig.2.1: Graph of ML techniques used in software projects.

Proposed Methodology

In this section, we will examine how machine-learning techniques can be used to assign tasks to team members without human intervention. The system will assign the work to the team members it achieves so by taking into account the following historical data about team members working on a project, which serves as crucial parameters to consider when to allocate tasks to specific roles. This data can be retrieved from open-source project management tools on GitHub or from any other private task management tool used by the organization.

3.1 DOD (Definition of Done):

In terms of project management and outputs, it is a method for delivering a high-quality product while also satisfying management or customers. To do this, only features that are actually completed are supplied, not only in terms of functionality but also in terms of quality. What's crucial to note here is that features are sometimes iterative, and there's always something to include or see, so understanding the "definition of done" is critical to ensure that the entire team is on the same page. As simple as it is to say, it is more difficult to do.

3.2 Commit History (CH):

A commit is an operation that adds the most recent modifications to the repository's source code, making these changes part of the repository's head revision. Unlike commits in a data management system, commits in a version control system are stored eternally in the repository. As a result, unless other users declare that they want to obtain a previous version of the source code in the repository, when they update or checkout from the repository, they'll get the most recent committed version. Version management systems make it simple to revert to previous versions. A commit within a versioning system is protected in this context because it can be rolled back at any time, even after it has been implemented.

3.3 Code Complexity (CC):

The term "programming difficulty" (or "software complexity") refers to a collection of qualities that any code can have. All of these qualities are concerned with how the code interacts with other code. The complexity of the code will be determined by the measurement of these features. It's a software quality rating for a piece of code. There are various methods for measuring complexity, but two, in particular, are often used: cyclomatic complexity and N-Path. In this section, we'll look at how to calculate code complexity, which is the most essential project attribute. Developers have been concentrating on how to assess code quality. There are a number of widely used tools for calculating technical debt (TD), which can be calculated as follows:

TD = (cost for fixing violations) + (cost for fixing duplications) + (cost to comment public API) + (cost of reducing the complexity below a threshold value) + (cost of fixing uncovered complexity)

Among all these factors mentioned above, code complexity is a major factor that is difficult to measure as it is influenced by many factors. According to McCabe cyclomatic complexity (MCC) was introduced by Thomas McCabe [23] in the year 1976. It measures the number of independent paths in graph theory. For example, if a method does not have the condition is MCC will be equal to 1 and for the program with many conditions MCC can be calculated as:

$$M= e – d+r$$

Where M is Mccabe cyclomatic complexity, e is a number of edges, d is a number of decision points, and r is a number of return statements in the graph.

3.4 Ted Backlogs (TB):

One of the most crucial parts of today's modern software development process is the backlog. It's a list of all the features and modifications that need to be applied to the product. The one thing that all of these things have in common is that they all have to provide value to the customer. User stories or actual features that need to be coded are common examples of these items. The list is prepared with the highest priority things at the top, and it can be updated and edited at any time during the event. Every item has an estimate of how long it will take to complete it next to it. Tasks that aren't directly related to merchandise development but are nevertheless necessary can be found in the backlog.

3.5 Team Calendar (TC):

A team calendar is a calendar that can be shared or viewed by everyone on the team. It will frequently display team meetings and block-out periods. Meetings for each team member, as well as individual work hours, availability, and planned time off, are frequently included. Team calendars are a wonderful way to encourage team transparency and visibility. They can assist in overcoming the difficulties that arise with collaboration, such as visibility, resource planning, and so on.

3.6 Ticket History (TH):

A ticket in any service desk software represents a work item that needs to be addressed by the team member. In Jira Software, it refers to a bug issue tracking, fixing the defect, and addressing clients’ requests.

The system considers a member's DOD, then checks the skills from the Commit History and judges the complexity of the code developed by the person, then determines coverage and cyclomatic complexity from Ted backlogs then checks the Team calendar for team member availability and finally determines the skills and aspirations of the team member from the Team Development Plan. The algorithm verifies all of these elements of a team member before deciding which assignment is most suited to their abilities. This information may be gleaned from the project logs that are kept for each project, and it can be used to rigorously train the model so that the next allocation system can make an appropriate decision without human intervention.

Fig: 3.1 Resource allocation Method

Conclusion

It has been determined from past project management research that software management solutions are utilized in the engineering phase of software development, to estimate needs, cost, effort, and risk, and machine learning methods are applied. However, there isn't a lot of research on machine learning algorithms in human resource allocation. Integer programming, Genetic algorithms, and Data Mining techniques are used to tackle the Resource Allocation problem. Much of the repetitive work of managers can be reduced if machine learning methods are utilized to train the model with existing project data. Optimizing the allocation procedure in the future can be focused on increasing the project's efficiency and productivity. From the results obtained it can be inferred that with ML algorithms we are obtaining less accuracy, in order to increase the model accuracy we can train the model with deep learning algorithms for further study.

References

[1] A. Idri, I. Abnane, and A. Abran, "Systematic mapping study of missing values techniques in software engineering data," in 2015.

[2] A. M. Alsalemi and E.-T. Yeoh, "A Systematic Literature Review of Requirements Volatility Prediction," in 2017.

[3] S. P. Pillai, S. Madhukumar, and T. Radharamanan, "Consolidating evidence-based studies in software cost/effort estimation—a tertiary study," in TENCON 2017-2017.

[4] P. Sharma and J. Singh, "Machine Learning-Based Effort Estimation Using Standardization," in 2018.
[5] J. Shivhare and S. K. Rath, "Software effort estimation using machine learning techniques,".
[6] Malhotra, R., & Chug, A. (2016). Software Maintainability: Systematic Literature Review and Current Trends. *International Journal of Software Engineering and Knowledge Engineering*, *26*(8), 1221–1253.
[7] Pillai, S. P., Madhukumar, S. D., & Radharamanan, T. (2017). Consolidating evidence-based studies in software cost/effort estimation-A tertiary study.
[8] Luis Daniel Otero, Grisselle Centeno, Alex J. Ruiz-Torres, Carlos E. Otero,A systematic approach for resource allocation in software projects Computers & Industrial Engineering, Volume 56, Issue 4,2009.
[9] Luis Daniel Otero, Grisselle Centeno, Alex J. Ruiz-Torres, Carlos E. Otero, A systematic approach for resource allocation in software projects, Computers & Industrial Engineering, Volume 56, Issue 4,2009. [10] C. E. Otero, L. D. Otero, I. Weissberger, and A. Qureshi, "A Multi-criteria Decision-Making Approach for Resource Allocation in Software Engineering," in 2010 12th International Conference on Computer Modelling and Simulation, Cambridge, United Kingdom, 2010, pp. 137–141, DOI: 10.1109/UKSIM.2010.32.
[11] J. Park, D. Seo, G. Hong, D. Shin, J. Hwa, and D.-H. Bae, "Human Resource Allocation in Software Project with Practical Considerations," Int. J. Softw. Eng. Knowl. Eng., vol. 25, no. 01, pp. 5–26, Feb. 2015, DOI: 10.1142/S021819401540001X.
[12] J. Xiao, Q. Wang, M. Li, Ye Yang, Fan Zhang, and Lizi Xie, "A constraint-driven human resource scheduling method in software development and maintenance process," in 2008 IEEE International Conference on Software Maintenance, Beijing, China, Sep. 2008, pp. 17–26, DOI: 10.1109/ICSM.2008.4658050.
[13] N. Bibi, A. Ahsan, and Z. Anwar, "Project resource allocation optimization using search-based software engineering — A framework," in Ninth International Conference on Digital Information.
[14] W. Aslam and F. Ijaz, "A Quantitative Framework for Task Allocation in Distributed Agile Software Development," IEEE Access, vol. 6, pp. 15380–15390, 2018, DOI: 10.1109/ACCESS.2018.2803685.
[15] M. Farhangian, M. Purvis, M. Purvis, and T. B. R. Savarimuthu, "Agent-Based Modeling of Resource Allocation in Software Projects Based on Personality and Skill," in Advances in Social Computing and Multiagent Systems, vol. 541, F. Koch, C. Guttmann, and D. Busquets, Eds. Cham: Springer International Publishing, 2015, pp. 130–146.
[16] T. J. A. Santos, A. M. Lima, C. A. L. Reis, and R. Q. Reis, "Automated support for human resource allocation in software process by cluster analysis," in Proceedings of the 4th International Workshop on Recommendation Systems for Software Engineering-RSSE 2014, Hyderabad, India, 2014, pp. 30– 31, DOI: 10.1145/2593822.2593830.

[17] S. Chakraverty, A. Sachdeva, and A. Singh, "A genetic algorithm for task allocation in collaborative software development using formal concept analysis," in International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014), Jaipur, India, May 2014, pp. 1–6, DOI: 10.1109/ICRAIE.2014.6909305.

[18] H. Y. Chiang and B. M. T. Lin, "A Decision Model for Human Resource Allocation in Project Management of Software Development," IEEE Access, vol. 8, pp. 38073–38081, 2020, DOI: 10.1109/ACCESS.2020.2975829.

[19] Yichuan Jiang and Jiuchuan Jiang, "Contextual Resource Negotiation-Based Task Allocation and Load Balancing in Complex Software Systems," IEEE Trans. Parallel Distrib. Syst., vol. 20, no. 5, pp. 641–653, May 2009, DOI: 10.1109/TPDS.2008.133.

[20] P. C. Pendharkar and J. A. Rodger, "An empirical study of the impact of team size on software development effort," Inf. Technol. Manag., vol. 8, no. 4, pp. 253–262, Nov. 2007, DOI: 10.1007/s10799-006-0005-3.

[21] Santos, T. J. A., Lima, A. M., Reis, C. A. L., & Reis, R. Q. (2014). Automated support for human resource allocation in software process by cluster analysis. 4th International Workshop on Recommendation Systems for Software Engineering, RSSE 2014-Proceedings, 30–31. https://doi.org/10.1145/2593822.2593830.

[22] Zhu, Q., & Ren, Z. (2010). Research on human resource configuration strategy in software engineering. Proceedings of the International Conference on E-Business and E-Government, ICEE 2010, 2764–2767. https://doi.org/10.1109/ICEE.2010.69

[23] Lamersdorf, A., & Rombach, D. (2009). Empirically-based decision support for task allocation in global software development. Proceedings-2009 4th IEEE International Conference on Global Software Engineering, ICGSE 2009, 281–284. https://doi.org/10.1109/ICGSE.2009.38

[24] A. Barreto, M. D. O. Barros, and C. M. L. Werner, ``Staf_ng a software project: A constraint satisfaction and optimization-based approach," Comput. Oper. Res., vol. 35, no. 10, pp. 3073_3089, Oct. 2008.

[25] G. Antoniol, M. Di Penta, and M. Harman., "Search-based techniques for optimization software project resource allocation." In Genetic and Evolutionary Computation–GECCO, Springer Berlin Heidelberg, January 19, 2004, pp.1425-1426.

[26] Z. Anwar., and A. Ahsan, "Comparative analysis of MOGA, NSGA-II and MOPSO for regression test suite optimization." International Journal of Software Engineering (IJSE), vol. 1, No. 7, pp. 41-56, January 2014.

[27] Q.Bai, "Analysis of particle swarm optimization algorithm." Computer and Information Science, CCSE, Vol. 3, No. 1, pp 180-184, Feb 2010.

[28] WN. Chen, and J. Zhang, "Ant colony optimization for software project scheduling and staffing with an event-based scheduler." IEEE Transactions on Software Engineering, Vol. 39, NO. 1, January 2013, pp 1-17.

[29] A. Lamersdorf and J. Münch, "A Survey on the State of the Practice in Distributed Software Development: Criteria for Task Allocation," in 2009 Fourth IEEE International Conference on Global Software Engineering.

[30] I. Richardson, V. Casey, D. Zage and W. Zage, "Global Software Development – the Challenges," University of Limerick, Ball State University: SERC Technical Report 278, Limerick, 2005.

[31] A. Fernández-del Viso Torre, A. Lamersdorf and J. Münch, "A Riskdriven Model for Work Allocation in Global Software Development Projects," in Sixth IEEE International Conference on Global Software Engineering, 2011.

[32] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, AddisonWesley Pub Co, 1989.

[33] André, M., Baldoquín, M.G., Acuña, S.T.: Formal model for assigning human resources to teams in software projects. Inf. Softw. Technol. 53(3), 259–275 (2011)

[34]. LePine, J.A., Buckman, B.R., Crawford, E.R., Methot, J.R.: A review of research on personality in teams: accounting for pathways spanning levels of theory and analysis. Hum. Resour. Manage. Rev. 21(4), 311–330 (2011)