2023 International Conference on Communication, Security and Artificial Intelligence (ICCSAI) | 979-8-3503-6996-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCSAI59793.2023.10421235

Optimizing Effort Estimation in Agile Software Development: Traditional vs. Advanced ML Methods

Neelam Sunda
Research Scholar, Research Centre
S. S. Jain Subodh College, RTU, Kota, India
Research.neelam@gmail.com

Ripu Ranjan Sinha
Research Supervisor, Research Centre
S. S. Jain Subodh College, RTU, Kota, India
drsinhacs@gmail.com

Abstract— Effort estimation is a fundamental procedure that involves forecasting the required time, resources, and effort needed to successfully accomplish a project, assignment, or activity. A major challenge for the software industry is accurate effort estimation. In recent years, the software industry has moved towards Agile Methodology since its development in 2001. The Agile methodology emphasizes iterative and incremental delivery, which allows the software to be delivered in smaller increments and provides greater flexibility in response to changes in requirements. Within an agile framework, effort estimation encompasses both traditional and advanced algorithmic methods. This study conducts a comparative analysis, evaluating traditional and advanced machine learning techniques for effort estimation. By analyzing several previous studies, the research identifies the most suitable techniques to achieve accurate estimations. Traditional effort estimation techniques often fall short in accurately predicting the efforts needed for a software project due to data limitations, lack of detail, neglect of complexity, unaccounted risks, and an inability to adapt to changes. The limitations of conventional approaches can be overcome by machine learning algorithms that can be trained on past data to identify hidden patterns and relationships and produce more accurate and efficient effort predictions.

Keywords—AGILE methodology, Software Effort Estimation, Machine Learning

I. INTRODUCTION

Software development success depends heavily on accurate effort estimation. It helps to plan and budget resources, manage risks, allocate resources, manage stakeholder expectations, and raise the overall quality of the final product. Most software-based businesses use Scrum, a popular Agile methodology, for the development of their projects, although there are certain problems with the technology's capacity to precisely measure software effort [53]. The project manager and development team face several challenges during effort estimation, including balancing scope and budget, managing stakeholder expectations, accounting for uncertainty, determining the right level of detail, and dealing with inconsistent data [1]. Overcoming these challenges requires a combination of experience, knowledge, and a flexible approach to estimation that is able to accommodate changes in requirements and stakeholder expectations. Traditional techniques of effort estimation are not always efficient in accurately predicting the efforts required for a software project. This is a result of a scarcity of information, an inadequate level of detail, a failure to consider complexity, an underestimation of risks, and an inability to adapt to change. In this day and age, software has taken a crucial role in most of systems due to the use of computers in every aspect of our lives. As a result, developing highly effective and sophisticated software is extremely important and a very challenging task [54]. Accurate effort estimation plays a crucial role in identifying and managing risks. Underestimating the effort required for a task can lead to missed deadlines and budget overruns. Conversely, overestimating the effort can result in unnecessary expenses, potentially impacting the overall project budget and timeline. Thus, precise estimation helps in accommodating changes and ensures efficient project planning and execution [2]. Incremental and iterative developments are fundamental parts of numerous well-known software development methodologies, for example, Scrum, Kanban, Extreme Programming (XP), Feature-driven development (FDD), and other Agile Software Development methods [3]. All agile strategies are created through iterative cycles and in incremental ways [4].

II. LITERATURE REVIEW

Alhamed et. al. [23], Planning Poker is a powerful technique for planning and size estimation. But Planning Poker depends on the availability of the experts team to create an estimate with consensus from all the experts. In this study, the author has implemented a crowdsourced Planning Poker model and extensively investigated the feasibility of applying crowdsourcing to Planning Poker for size estimation. The study has conducted 80 Crowd Planning Poker (CPP) rounds and 807 estimates were received from the crowd instead of a team of experts. Unlike the traditional Planning Poker method where there were limited members for estimation, there is a larger set of crowd workers who provide their estimation in CPP. Fleiss Kappa method is used to calculate consensus for a large crowd. Out of the 30 trials conducted, crowd workers correctly predicted 7 trails, whereas the experts correctly estimated 14 trials. However, the estimation performance is similar if the analysis is based on which prediction was most accurate for each trail.

P. Sudarmaningtyas et. al. [24], In this study, the authors proposed a modification of Planning Poker to mitigate the weakness of Planning Poker. The estimation method and consensus procedure of the Planning Poker are modified by the suggested model. Subjectivity of estimators is reduced by modifying the estimation process, whereas in consensus process is shifted from traditional to automation. There were no rules for size estimation in the Planning Poker, but with this paper the author proposed a model which can determine the effort based on certain variables. Changes made in the consensus process helps to accelerate time estimation.

T. J. Gandomani et. al. [25], Aim of the authors in this paper is to evaluate the precision of Planning Poker using consensus and averaging. The author monitored a software company for two years, where the company employed either consensus-based Planning Poker or average-based Planning Poker for cost estimation. The paper looks at the performance of 29 completed projects and measures the error between predicted estimation and actual size using Absolute Error and Relative Error. The findings show that when the company used averaging for size estimation there was a 9% error whereas, there is a 3% error when consensus is used to estimate a User Stories' size.

Alhazmi et. al. [26], Scrum is a widely used Agile software development process that is covered in-depth by the author. Scrum encourages incremental, quick deliveries and iterative feedback from clients. A crucial step in the Scrum methodology is sprint planning, where the team decides on the goals for an upcoming release and develops a plan to reach those goals. During the sprint planning process, managers can use the Sprint Planning Decision Support System (SPESS), a tool, to help them make educated decision. It provides valuable insights and data-driven recommendations to aid in effective Sprint planning and resource allocation. Among other factors in Sprint planning, SPESS prioritizes engineer skills, designer status, and task interdependence. Its outcomes ensure that each team member's tasks during each Sprint align with their expertise, maximizing their contributions. Additionally, the tool optimizes project planning for the shortest possible time frame, leading to efficient project execution.

Faria et. al. [27], The author discusses the consequences of relying on expert judgment for effort estimation in a mid-sized software firm. The findings highlight inconsistencies in estimates, resulting in cost overruns and project delays. Limited use of historical data, challenges in decision-making, and the potential for improvement through data-driven approaches are also addressed.

Prasada et. al. [28], provided an approach using three ML techniques: RBFN, GRNN, and Adaptive Neuro-Fuzzy Modelling (ANFM) along with the story-point approach. Data included in the data set is obtained from 21 projects. Story Points and Project Velocity were supplied as input in anfis, newgrnm, newrb and newrbe functions. Furthermore, these functions were applied to the training and testing data set, and MMRE, MMER, and PRED were used to evaluate the result accuracy.

Alsaadi et. al. [29], present a systematic literature review of data-driven techniques for estimating the time needed to create a user story. The author considered five perspectives during the comprehensive overview, which include data-driven techniques, numerous performance measurement techniques, the accuracy of estimation, independent factors affecting estimation, and dataset characteristics. On the basis of 11 studies, the author claims that data-driven techniques are more accurate in estimating the effort required for user stories.

Alloghani et. al. [30], The rapid development of machine learning, which is similar to ideas like Big Data and data science, is revealed via a review of papers published from 2015 to 2018. The decision tree, support vector machine, and Naive Bayes algorithms were found to be the most often used supervised learners by the authors, while k-means, hierarchical clustering, and principal component analysis were found to be the most widely used unsupervised learners. Due to ongoing advancements in machine learning and data science, the study also highlighted other well-known algorithms, such as ensembles and reinforcement learning, proposing that future systematic reviews can focus on the same.

TABLE I. COMPARATIVE STUDY ON THE PREVIOUS WORK DONE

Ref no	Year	Used Technique	No. of data set	Description
[23]	2021	Planning Poker	301	Tested data in total 9 trials with total data of 301 in which accepted data is 121, Ambiguous data is 70 and rejected data is 110. MRE in one week is 17% and in Two week is 3%.
[24]	2020	Planning Poker	38	Three additional criteria were added to the estimating procedure, and each factor was given a weight and score.
[25]	2019	Planning Poker	537	Total estimated size is 537 in 29 trails and the actual size is 533.5 with absolute error is 16.5 and Relative error is 0.03.
[26]	2018	Planning Poker, Hungarian Algorithm	36	Proposed a novel technique, called Sprint Planning Decision Support System (SPESS) which helps managers plan better for a sprint. Two experiments are performed in this study. In the first experiment with 20 tasks, the completion of the sprint using the SPESS tool is 20.02 days. In the second experiment, the sprint which used the SPESS tool required 17.6 days, whereas the sprint without SPESS required 18.0 days.

[27]	2012	Expert Judgment	1000	Done the experiment in 15-15 of two groups and get the response rate is 47%. at the 1000 staff days get the mean 319, standard deviation 693 and coefficient of variance is 2.17.
[28]	2018	Story Point and Project Velocity	21	In this, 21 projects data set were evaluated using 3 ML techniques: ANFIS, RBFN, GRNN. As performance metrics MMRE, MMER and PRED were used.
[29]	2022	Data-driven Techniques	11	In this, twenty data-driven algorithms were identified. Frequently used algorithms were Decision Tree, Support Vector Machines and Bayesian Networks.

III. Findings and analysis

Here is a thorough analysis of studies obtained from reviewing the 8 papers in the previous section out of the 8 papers, one of the papers [29] is a journal article, another one[30] is a book chapter, while the rest are from conference proceedings. Majority of the papers selected were published in the last five years (7 out of the 8), half of it (4 out of the 8) were published in the last three years.

[redacted]*RQ1: In Software Development, which type of methods are better for effort estimation?*

The aim of this question was to identify which method is better for effort estimation, traditional or ML-based. This will help companies as well as researchers to get a better overview of the accuracy of each technique to calculate the effort estimation of a software project. There was a total of 8 techniques that were identified in our study. Out of the 8 techniques, 5 of the techniques were traditional effort estimation methods, namely Sprint Planning Decision Support System, Consensus Planning Poker, Crowd Planning Poker, Extended Planning Poker, and Averaging Planning Poker. The remaining methods are ML-based effort estimation. ML algorithms used for estimation are ANFIS, GRNNs and SVM. Given enough historical data, ML-based methods can find patterns that might have been overlooked in traditional methods. Also, for ML-based algorithms, as we get more data regarding effort estimation, the better it will adapt to the data.

[redacted]*RQ2: When should we select one technique over another for effort estimation?*

This question aims to discuss which technique would be most effective for a given software project. Though ML-based methods are generally better than traditional methods, they have their own limitations. ML-based methods are effective when there is a good amount of quality data available. If we do not have adequate data to properly train ML models then the accuracy of estimation will drop significantly. Another limitation when traditional methods would be better, is when a project is simple. For simple projects, both traditional and ML-based would have the same accuracy, but traditional methods would be faster and more reliable. Other limitations for ML-based would-be resource restraints, or if the project has a low risk tolerance.

IV. Methodology

[redacted]*Effort Estimation in Agile Software Development*

Various methods for estimating efforts in agile software development were presented in previous years, for example, analogy, story points, planning poker, and expert judgment [5-8]. Implementing machine learning can help in effort estimation by providing more accurate and efficient predictions, automating the estimation process, providing greater flexibility, improving consistency, and improving risk management. By leveraging the power of machine learning, product owners can overcome some of the challenges associated with traditional techniques of effort estimation and deliver software products more effectively and efficiently. A branch of artificial intelligence known as machine learning (ML) enables computers to automatically learn from experience and advance without explicit programming.[9]. ML algorithms are designed to learn from data and make predictions based on that information. On effort estimation using ML approaches, many research efforts have been carried out. [10-11].

[redacted]*Machine Learning Algorithms*

By analyzing statistical models and methods, machine learning enables computer systems to make predictions and judgments without the use of explicit programming. The process of machine learning involves both inference and training to enable computers to learn from data and improve their performance over time, enabling them to handle complex tasks and adapt to changing environments. The majority of AI work in recent years has been driven by ML techniques [12-13].

The inference phase, alternatively referred to as the testing or prediction phase, involves the utilization of the trained machine learning model to generate predictions or make choices regarding novel, unobserved data. After being trained on a dataset with labels, the model acquires knowledge of patterns and correlations within the data, enabling it to be utilized for its designated purpose. During the inference phase, the model utilizes input data to generate output predictions without making any adjustments to its parameters [14]. The primary goal is to employ the trained model in order to generate precise and dependable forecasts in practical situations. The training phase encompasses the iterative process through which the machine learning model acquires knowledge from the annotated training data, hence enhancing its efficacy in accomplishing the designated goal [15]. The model goes through parameter adjustments during the training phase by considering the input data and their accompanying labels in the case of supervised learning or the data properties in the case of unsupervised learning. In supervised learning, the optimization method seeks to minimize the loss between model predictions and real labels; in unsupervised learning, it seeks to identify the most optimal representations. Before training the model on a dataset, we need to perform various steps like data preprocessing, feature selection, etc [49].

[redacted] *Significant Approaches of Machine Learning*

Machine learning (ML) encompasses different significant approaches, such as supervised learning with labeled data, unsupervised learning for the purpose of pattern identification, and reinforcement learning that involves interaction and feedback. Supervised learning is a technique that entails training a machine learning model using labeled data, where each input data point is paired with a corresponding target or label. In contrast, unsupervised learning pertains to the analysis of data that lacks explicit labeling or predetermined aims. Reinforcement learning is a field of machine learning that focuses on teaching an autonomous agent to make decisions within a given environment in order to accomplish predetermined objectives. When working with any dataset, there are various problems that need to be resolved before it can be used. A significant problem is the issue of data imbalance, where one category has disproportionally large data compared to others. This can be resolved by applying techniques like outlier detection on this majority category, and reducing the category size to be comparable size [52]. Accuracy of a ML-based model highly depends upon the data on which its trained on. Methods like feature selections not only reduce the size of the dataset, which in turn leads to lower training time and better accuracy [51]. Figure 1 presents the categorization of these algorithms.

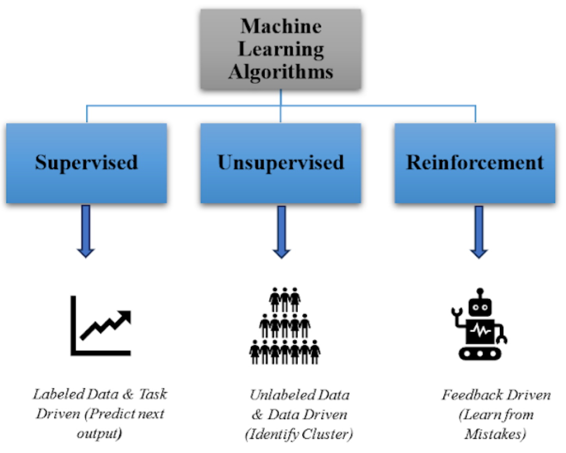
Machine Learning Algorithms
Supervised
Unsupervised
Reinforcement
Labeled Data & Task Driven (Predict next output)
Unlabeled Data & Data Driven (Identify Cluster)
Feedback Driven (Learn from Mistakes)

Fig. 1. Classification of Machine Learning Algorithms

1) Supervised Machine Learning:

Supervised learning is commonly applied for data classification, requiring a dataset where each input is associated with a labeled output. Classification uses a model which is trained on labeled data to predict the result of unknown instances [50]. Various models can be developed to predict results from data source. The machine learns from this dataset to create an inference function that maps each input to the correct output [16].

Through supervised learning, the machine receives precise labeling of input data, allowing it to utilize the inference function effectively. For example, a system subjected to labeled tiger photographs can pick up on distinctive tiger traits. Subsequently, it can utilize this acquired knowledge to detect new and previously unidentified tiger images [17].

2) Unsupervised Learning:

In unsupervised learning, the machine processes a dataset without the presence of labels. It organizes the input data into clusters based on similarities, without assigning them to specific groups. The main goal of unsupervised learning is to unearth underlying structures or patterns in each set of data.[18]. For example, in unsupervised learning, the machine can automatically group similar images without any prior knowledge of the specific group names or categories. It identifies common patterns or similarities within the images and clusters them accordingly, revealing meaningful associations within the data.

3) Reinforcement Learning:

Reinforcement learning is a goal-oriented method of machine learning. Through feedback learning, whether it takes the form of negative or positive reinforcement, the machine tries to accomplish its goal. It can employ reinforcement learning, for instance, to master the game of Pac-Man by maximizing its score and gradually enhancing its gameplay abilities. In addition, semi-supervised learning techniques combine a sizable amount of unlabeled data with a limited amount of labeled data. This approach is particularly valuable for applications that involve a high cost of labeling information. By leveraging the available labeled data together with the abundant unlabeled data, semi-supervised learning can achieve more accurate and cost-effective results in various scenarios [19]. A significant distinction among these processes lies in the data requirements and training environments each technique necessitates. Supervised learning relies on a comprehensive labeled dataset, while unsupervised learning requires a vast amount of unlabeled data without the need for data labeling. Reinforcement learning stands apart from other learning methods as it does not rely on a dataset for training. Instead, it necessitates a training environment to conduct experiments, utilizing a clearly defined metric for optimizing data. Through trial and error in this training atmosphere, the machine learns to maximize rewards and optimize decision-making processes without explicitly being provided with a dataset for learning.

[redacted] *Agile Methodology*

In recent years, the software industry has moved towards Agile methodology since its development in 2001. The primary goal is to optimize customer satisfaction while also delivering the specified software on time. Agile project management has become the most popular approach to software development because of its adaptability, flexibility, and focus on customer feedback [22]. Agile project management isn't just one framework; rather, it can be used as an umbrella term to refer to a variety of frameworks. Scrum, Kanban, Feature-driven development, and Adaptive Project Frameworks are examples of prevailing frameworks associated with agile project management [8]. In Agile techniques, a project is broken up into several phases, and these phases are used to oversee the entire development process. The framework required constant stakeholder participation and continuous development at every stage. After the project development process begins, the development teams cycle through a process of planning, carrying out, and evaluating their work [23]. An agile team produces work in manageable, short chunks referred to as increments.

1) Estimation Practices in Agile Technology:

Any project has the risk of running late and overspending. However, by using the right estimating methods, the project owner can significantly decrease the chances of such an occurrence. Agile estimation measures the amount of time needed to finish a task with the highest priority on the product backlog [8][23]. In Agile development procedures, the project owner determines each developer's responsibilities by predicting the time needed to complete a particular activity. Work is then distributed to each person in accordance with the predicted time. In order to correctly organize the sprints, the product owner estimates the amount of time it will take to execute a user story. A sprint, as the name implies, is a brief period of time, usually 2 to 4 weeks, that specifies the time needed to complete a work [22, 26]. Numerous studies have been conducted in recent years to identify the most popular agile estimate methodologies: Planning-Poker (numbering cards to estimate the user story), Expert Judgement, Analogy (relative sizing approach) [22-23, 25], Delphi estimation, use case points, etc. [28].

The present estimation techniques are less complex and expert-oriented. Although these methods are critiqued, they must always be used in conjunction with expert judgment, and the skill of the expert determines the method's accuracy and dependability [27]. So, there is a need for data-intensive algorithmic techniques that are unbiased and more accurate in generating results.

[redacted]*Technologies*

1) Radial Basis Function Network

Radial Basis Function Network (RBFN) is a special type of artificial neural network where we use the Radial Basis Function (RBFs) as the activation function. RBFN is based on the idea of a biological receptive field, where function mapping is performed by the network by employing a local receptive field [36]. RBFN is a feed-forward network made up of three parts:

Input Layer: The input layer is the first layer of the RBFN, which receives numeric values representing various factors related to software development as input features. These input features may include factors like project complexity, line of code, etc.

Hidden Layer: The hidden layer is made up of a set of RBFs. RBFs are used to compute the similarity between the input data and the center point. The hidden layer's job is to convert the data from the input layer into a higher-dimensional space. This layer's computation can be written mathematically as:

$$\phi_i = e^{-\frac{\left\| \underline{X} - \underline{u_i} \right\|^2}{2\,\sigma_i^2}}$$

Output Layer: In the output layer, the linear activation function is employed for both classification and regression. Computation in this layer is achieved by a weighted sum of data received from the previous layer.

$$y = \sum_{1}^{n} \omega_i \, \phi_i$$

2) General Regression Neural Network

General Regression Neural Network (GRNNs) are a feedforward neural network that is used for regression, prediction, or classification. It is a type of RBFNs. In 1991, D.F. Specht suggested GRNNs.

GRNNs consist of four layers, namely, 1. input layer, 2. pattern layer, 3. summation layer, and 4. output layer [37]. The first layer receives all the input features and passes all the features to each neuron in the pattern layer. The pattern layer uses a nonlinear function, usually an exponential function as its activation function. Each pattern layer unit is assigned to a single cluster center. The summation layer receives these values as input. In the summation layer, a signal vector from the previous layer and a weighted vector are used to create a dot product in each unit. In the output layer, the value received from the summation layer is simply divided by to get the desired estimated value.

GRNNs estimation can be mathematically represented as:

$$\hat{Y}(X) = \frac{\sum_{i=1}^{m} A_i exp(-\frac{D_i^2}{2\sigma^2})}{\sum_{i=1}^{m} B_i exp(-\frac{D_i^2}{2\sigma^2})}$$

3) Adaptive Neuro-Fuzzy Modeling

ANFIS, or Adaptive Neuro Fuzzy Inference System, is a hybrid of fuzzy logic and neural networks. A feedforward neural network with supervised learning is an adaptive network. It is a feedforward network with adaptive nodes, which implies that each node's output is determined by the parameters.

Fuzzy inference systems (FIS) consist of five functional blocks [38]:

- a set of fuzzy if-then rules in a rule base.

- In a database that is utilized by fuzzy rules, a membership function of the fuzzy set is defined.
- The rule inference processes carried out by a decision-making unit.
- an interface for fuzzification that transforms the sharp inputs into linguistic value levels.
- a defuzzification interface that transforms the interface's fuzzy data back into clear, distinct data.

Consider a FIS taking two variables x and y as input and producing z as an output. Then the layers for ANFIS are a follows [39]:

Layer 1: Layer 1 consists of adaptive nodes with each having a function

$O_{i,i} = \mu_{B_{i-2}}(x)$, for i = 3, 4 or

$O_{1,i} = \mu_{A_i}(x)$, for i = 1, 2

Here, and are linguistic labels associated with each node.

Layer 2: In layer 2, each node outputs the product of all signals.

$O_{2,i} = \mu_{A_i}(x)\mu_{B_i}(y)$ for i = 1,2

Layer 3: Here, each node i computes the ith rule's normalized firing strength. Normalized firing strength calculates the ratio of ith node's firing strength to total firing strengths

$$O_{3,i} = \bar{w_i} = \frac{w_i}{w_1 + w_2}$$

Layer 4: Here, nodes are adaptive node with node function

$$O_{4,i} = \bar{w_i} f_i = \bar{w_i}(p_i x + q_i y + r_i$$

Layer 5: This is an output layer, where the unit calculates the final output by adding all previous layer values

$$O_{5,1} = \sum_i \bar{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$$

4) Support Vector Machine

The support vector machine (SVM) is a computational model that is utilized for both classification and regression tasks. This algorithm is based on finding a hyperplane which divides the dataset into two sets by aiming to widen the boundaries. In the field of machine learning, a classifier refers to a constructed model that is designed to draw conclusions regarding a specific class based on the presence of supplementary data. The term "classification" pertains to the process of assigning a distinct value to an unlabelled record. SVM variants are SVRs. Regression concerns become classification challenges. When training SVM, a linear decision surface (hyperplane) splits two vector segments. The margin between the LDS and the vectors that are nearest to it is maximum in this separation. These entities are commonly referred to as support vectors. The input vectors undergo a non-linear mapping process to transform them into a high-dimensional space, as required by the non-linear LDS. The feature space is utilized to construct the Linear Discriminant Analysis (LDA) model by selecting characteristics that optimize the margin, resulting in a minimized generalization error for the machine. This LDS has the most comprehensive generalization among all speculative hyperplanes. The utilization of an optimal hyperplane enhances the prediction performance of classification or regression models [40].

$$f(x) = \omega T\, \phi(x) + b$$

where x is the input vector, $\phi(x)$ is the feature mapping function, ω is the weight vector, and b is the bias term.

V. Research Methodology

This section discusses the comprehensive analysis and how the exploration of all the studies mentioned in section 3 are conducted.

[redacted]*Research Questions*

RQ1: In Software Development, which type of methods are better for effort estimation?

The question identifies which method is better, traditional or ML-based method, as well as compare the accuracies of each technique.

RQ2: When should we select one technique over another for effort estimation?

This question focuses on what parameters a company should look at before selecting a technique.

[redacted]*Search Strategy*

Using a list of keywords as the search string, the data sources from which studies are to be drawn are determined.
For tradition methods, the search is:
("effort estimation" || "cost estimation") && (Scrum || Agile || "Planning Poker")
For ML-based methods, the search is:
("effort estimation" || "cost estimation") && (Scrum || Agile || "Planning Poker") && ("machine learning" || "deep learning" || "artificial intelligence")

[redacted] *Literature Review*

The journal or proceedings selected for literature review are published between 2012 and 2022. Also, all the journals or proceedings must be in English and have valid DOI. Papers from the search are from the five resources shown in Table 1. IEEE Xplore has the highest contribution out of the five resources.

TABLE II. Paper Distribution

No.	Source Name	URL	Articles
1.	IEEExplore	https://ieeexplore.ieee.org/	5
2.	Wiley	onlinelibrary.wiley.com/	0
3.	Springer	link.springer.com/	3
4.	ACM	https://dl.acm.org/	0
5.	ScienceDirect	www.sciencedirect.com/	0

Total	8

VI. Conclusion

In an agile framework, the role of the product owner is pivotal in the procedure of successfully delivering a product. This responsibility includes effort estimation, which is a cornerstone of a successful project planning and execution. In our study, we have seen both traditional as well as ML-based effort estimation methods, each presenting their own pros and cons.

Traditional effort estimation methods, like consensus Planning Poker, expert judgment, etc. lean heavily on the expertise of the team members. Traditional methods are very reliant on the availability and subjective input for the team members. But on the other hand, they provide a quick estimation for simpler projects.

ML-based effort estimation techniques use historical data of effort estimation in previous projects along with multiple statistical models and algorithms, to produce a more effective effort estimation. Unlike traditional methods, ML-based methods keep on improving as the dataset is increased and improved. So, if we have a large set of historic data then ML-based methods yield a more accurate prediction as compared to traditional methods.

In this study, we conduct analysis of various methods outlined in the existing literature for effort estimation. Our findings highlight those traditional techniques, while quick and straightforward, may fall short in delivering the same level of accuracy as their advanced counterparts. Advanced machine learning techniques, despite their inherent complexity, hold the potential to provide significantly more accurate predictions and streamline the estimation process.

The project's nature, the desired level of accuracy, and the availability of pertinent data all play major roles in the decision between traditional and modern procedures. To choose the best course of action for their unique set of circumstances, project managers and product owners have to thoroughly evaluate various factors, keeping in mind that the accuracy of effort estimating can have a significant impact on project success.

Reference

[1] B. W. Boehm, Software Engineering Economics, Prentice Hall, 1981.

[2] J. M. Bhat, M. Gupta and S. N. Murthy, Overcoming requirements engineering challenges: Lessons from offshore outsourcing, IEEE Softw. 23(1), pp. 38–44, 2006.

[3] A. Cockburn, Using both incremental and iterative development (2019), SWEN 256: Software Process and Project Management,

[4] N. B. Moe and T. Dingsøyr, "Scrum and team effectiveness: Theory and practice, in Agile Processes in Software Engineering and Extreme Programming", Springer Lecture Notes in Business Information Processing, Vol. 9, pp. 11–20, 2008.

[5] Bahlerao S, Ingle M, "Generalized agile estimation method", International Journal Advance Science Engineering Information Technology 1(3), pp. 262–267, 2011.

[6] Coelho E, Basu A, "Effort estimation in agile software development using story points", International Journal Appl Inform System (IJAIS) 3(7), 2012.

[7] Mahnicˇ V, Hovelja T, "On using planning poker for estimating user stories", Journal of System Software, 85(9), pp. 2086–2095, 2012.

[8] Munialo SW, Muketha GM, "A review of agile software effort estimation methods", International Journal of Computer Application and Technology Research 5(9), pp. 612–618, 2016.

[9] Das S, Dey A, Pal A, Roy N, "Applications of artificial intelligence in machine learning: review and prospect", International Journal of Computer Application 115(9), 2015.

[10] Braga PL, Oliveira AL, Meira SR, "Software effort estimation using machine learning techniques with robust confidence interval", 7th International conference on hybrid intelligent systems (HIS 2007), pp 352–357, 2007.

[11] Sree SR, Rao P, Mounika M, "An early stage software effort estimation in agile methodology based on user stories using machine learning techniques", International Journal of Research Application in Science Engineering Technology (IJRASET) 5 (XII), 2017.

[12] O. Varol , E. Ferrara , C. Davis , F. Menczer , A. Flammini, "Online human-bot interactions: Detection, estimation, and characterization", in: International AAAI Conference on Web and Social Media, pp. 280–289, 2017.

[13] Priya Gour, Sudhanshu Vashistha and Pradeep Jha, "Twitter Sentiment Analysis Using Naive Bayes based Machine learning Technique", 2nd International Conference on Sentiment Analysis and Deep Learning (ICSADL 2022), Springer - Advances in Intelligent Systems and Computing Series, 2022.

[14] Rahul Misra and Dr. Ramkrishan Sahay, "A Review on Student Performance Predication Using Data Mining Approach", International Journal of Recent Research and Review, Vol. X, Issue 4, pp. 45-47, December 2017.

[15] M. Ji, Y. Sun, M. Danilevsky, J. Han, and J. Gao, "Graph regularized transductive classification on heterogeneous information networks," in Proc. Eur. Conf. Mach. Learn. Knowl. Discovery Databases, Part I, pp. 570–586, 2010.

[16] Vipin Singh, Manish Choubisa and Gaurav Kumar Soni, "Enhanced Image Steganography Technique for Hiding Multiple Images in an Image Using LSB Technique", TEST Engineering & Management, Vol. 83, pp. 30561-30565, 2020.

[17] Bartmann, D., Bakdi, I. & Achatz, M., "On the design of an authentication system based on keystroke dynamics using a predefined input text, Techniques and Applications for Advanced Information Privacy and Security", Emerging Organizational, Ethical, and Human Issues 1(2): 149, 2007.

[18] S. Khalid, T. Khalil, and S. Nasreen, "A survey of feature selection and feature extraction techniques in machine learning", in Proc. Sci. Inf. Conf., pp. 372-378, Aug. 2014.

[19] L. Deng and D. Yu, "Deep learning: Methods and applications", Found. Trends Signal Process, vol. 7, nos. 34, pp. 197-387, Jun. 2014.

[20] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", Nature, vol. 521, no. 7553, p. 436, 2015.

[21] Sridhar, Arjun, and J. S. Rajshekhar. "AI-Integrated Proctoring System for Online Exams", Journal of Artificial Intelligence 4, no. 2, pp. 139-148, 2022.

[22] Manju Vyas, Amit Bohra, Dr. C.S. Lamba and Abhilasha Vyas, "A Review on Software Cost and Effort Estimation Techniques for Agile Development Process", Manju Vyas et al. International Journal of Recent Research Aspects, Vol. 5, Issue 1, pp. 1-5, March 2018.

[23] M. Alhamed and T. Storer, "Playing Planning Poker in Crowds: Human Computation of Software Effort Estimates," 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), pp. 1-12, 2021.

[24] P. Sudarmaningtyas and R. B. Mohamed, "Extended Planning Poker: A Proposed Model," 2020 7th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), pp. 179-184, 2020.

[25] T. J. Gandomani, H. Faraji and M. Radnejad, "Planning Poker in cost estimation in Agile methods: Averaging Vs. Consensus," 2019 5th

Conference on Knowledge Based Engineering and Innovation (KBEI), 2019, pp. 066-071, 2019.

[26] A. Alhazmi and S. Huang, "A Decision Support System for Sprint Planning in Scrum Practice," SoutheastCon 2018, pp. 1-9, 2018.

[27] P. Faria and E. Miranda, "Expert Judgment in Software Estimation During the Bid Phase of a Project -- An Exploratory Survey," 2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement, pp. 126-131, 2012.

[28] Prasada Rao, C., Siva Kumar, P., Rama Sree, S., Devi, J. (2018). An Agile Effort Estimation Based on Story Points Using Machine Learning Techniques. In: Bhateja, V., Tavares, J., Rani, B., Prasad, V., Raju, K. (eds) Proceedings of the Second International Conference on Computational Intelligence and Informatics. Advances in Intelligent Systems and Computing, vol 712. Springer, Singapore. https://doi.org/10.1007/978-981-10-8228-3_20

[29] Alsaadi, B., Saeedi, K. Data-driven effort estimation techniques of agile user stories: a systematic literature review. Artif Intell Rev 55, 5485–5516 (2022). https://doi.org/10.1007/s10462-021-10132-x

[30] Alloghani, M., Al-Jumeily, D., Mustafina, J., Hussain, A., Aljaaf, A.J. (2020). A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science. In: Berry, M., Mohamed, A., Yap, B. (eds) Supervised and Unsupervised Learning for Data Science . Unsupervised and Semi-Supervised Learning. Springer, Cham. https://doi.org/10.1007/978-3-030-22475-2_1

[31] M. Tsunoda, A. Monden, J. Keung and K. Matsumoto, "Incorporating Expert Judgment into Regression Models of Software Effort Estimation," 2012 19th Asia-Pacific Software Engineering Conference, pp. 374-379, 2012.

[32] C. Jie, "Varying-step Correlation Analysis Identification Algorithm Based on the Modified Fibonacci Sequence," 2011 Third International Conference on Measuring Technology and Mechatronics Automation, pp. 1105-1108, 2011.

[33] R. Kurnia, R. Ferdiana and S. Wibirama, "Software Metrics Classification for Agile Scrum Process: A Literature Review," 2018 IEEE International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), pp. 174-179, 2018.

[34] P. L. Ayunda and E. K. Budiardjo, "Evaluation of Scrum Practice Maturity in Software Development of Mobile Communication Application", IEEE 2020 3rd International Conference on Computer and Informatics Engineering (IC2IE), 2020.

[35] H Yuliansyah, S N Qudsiah, L Zahrotun and I Arfiani, "Implementation of use case point as software effort estimation in Scrum Framework", IOP Conf. Series: Materials Science and Engineering 403, pp. 1-9, 2018.

[36] J. Moody and C. Darken, "Fast Learning in Networks of LocallyTuned Processing Units," Neural Computing, vol. 1, pp. 281-294, 1989.

[37] D. F. Specht, "A general regression neural network," in IEEE Transactions on Neural Networks, vol. 2, no. 6, pp. 568-576, Nov. 1991, doi: 10.1109/72.97934.

[38] J. . -S. R. Jang, "ANFIS: adaptive-network-based fuzzy inference system," in IEEE Transactions on Systems, Man, and Cybernetics, vol. 23, no. 3, pp. 665-685, May-June 1993, doi: 10.1109/21.256541.

[39] Jang, Sun, Mizutani (1997) – Neuro-Fuzzy and Soft Computing – Prentice Hall, pp 335–368, ISBN 0-13-261066-3

[40] R Mizanur et.al., "Software Effort Estimation Using Machine Learning", International Journal of Advanced Computer Science and Applications, Vol. 14, No. 4, 2023.

[41] Sharma, Sudhir, and Shripal Vijayvargiya. "An optimized neuro-fuzzy network for software project effort estimation." IETE Journal of Research (2022): 1-12.

[42] Harish Kumar, K., and K. Srinivas. "Efficient Analogy-based Software Effort Estimation using ANOVA Convolutional Neural Network in Software Project Management." Smart Intelligent Computing and Applications, Volume 1: Proceedings of Fifth International Conference on Smart Computing and Informatics (SCI 2021). Singapore: Springer Nature Singapore, 2022.

[43] Tashtoush, Yahya M., et al. "Project management effort estimation using agile manager game platform." 2022 13th International Conference on Information and Communication Systems (ICICS). IEEE, 2022.

[44] Song, Liyan, and Leandro L. Minku. "Artificial Intelligence in Software Project Management." Optimising the Software Development Process with Artificial Intelligence. Singapore: Springer Nature Singapore, 2023. 19-65.

[45] Rashid, Chaudhary Hamza, et al. "Software Cost and Effort Estimation: Current Approaches and Future Trends." IEEE Access (2023).

[46] Shukla, Suyash, and Sandeep Kumar. "Self-Adaptive Ensemble-based Approach for Software Effort Estimation." 2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, 2023.

[47] Ünlü, Hüseyin, et al. "Utilization of Three Software Size Measures for Effort Estimation in Agile World: A Case Study." 2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE, 2022.

[48] Wakudekar, Sachin, et al. "Design of A Novel Model that Aids Task Estimation in Comparison with a Regression Tree." 2023 10th International Conference on Computing for Sustainable Global Development (INDIACom). IEEE, 2023.

[49] Rahul, V. Raj and Monika, "Sentiment Analysis on Product Reviews," 2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 2019, pp. 5-9, doi: 10.1109/ICCCIS48478.2019.8974527.

[50] Rahul, H. Bansal and Monika, "Classification Techniques Used in Sentiment Analysis & Prediction of Heart Disease using Data Mining Techniques: Review," 2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), Ghaziabad, India, 2019, pp. 1-6, doi: 10.1109/ICICT46931.2019.8977707.

[51] Rahul, Priyansh Kedia, Subrat Sarangi & Monika (2020) Analysis of machine learning models for malware detection, Journal of Discrete Mathematical Sciences and Cryptography, 23:2, 395-407, DOI: 10.1080/09720529.2020.1721870

[52] N. Khanna, O. Agarwal and Rahul, "Software Change Prediction using Ensemble Learning on Object Oriented Metrics," *2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, India, 2022, pp. 1369-1373, doi: 10.1109/ICICCS53718.2022.9788196.

[53] N. . Sunda and R. R. . Sinha, "A Review: Effort Estimation Model for Scrum Projects using Supervised Learning", *IJRITCC*, vol. 11, no. 11s, pp. 302–308, Oct. 2023.

[54] N. . Gupta, R. R. . Sinha, A. . Goyal, N. . Sunda, and D. . Sharma, "Analyze the Performance of Software by Machine Learning Methods for Fault Prediction Techniques", *IJRITCC*, vol. 11, no. 5s, pp. 178–187, May 2023.