Interplay of Machine Learning and Software Engineering for Quality Estimations

Hamza Abubakar*, M.S. Obaidat, *Fellow of IEEE and Fellow of SCS* †,
Aaryan Gupta‡, Pronaya Bhattacharya§, Sudeep Tanwar, *Member, IEEE*¶
*‡§¶Department of Computer Science and Engineering, Institute of Technology, Nirma University, Ahmedabad, Gujarat, India
† Dean and Professor, College of Computing and Informatics, University of Sharjah, Sharjah 27272, UAE,
with King Abdullah II School of Information Technology, University of Jordan, Amman 11942, Jordan
and with University of Science and Technology Beijing, Beijing 100083, China.
Email: *habubakar89@gmail.com, †m.s.obaidat@ieee.org, ‡gupta.aaryan8@gmail.com,
§pronoya.bhattacharya@nirmauni.ac.in, ¶sudeep.tanwar@nirmauni.ac.in

***Abstract*—In this era, the agile mindset has innovated the traditional software engineering (SE) process through the integration of *DevOps* flow engines, scrum iterations, and automation of continuous integration (CI) and continuous deployment (CD) cycles. However, the CI/CD integration requires manual code-revisions and refactoring at large scales. Recently, machine-learning (ML) is employed in SE that allows legacy codes to be highly dynamic, less coupling in related modules, automatic code versioning, and refactoring, with less coupling among related modules. However, over time, ML models tend to become bulky, with increasing monotonic losses during model training. To address this, SE techniques like code revisions are employed over ML codes to allow low-order training losses, that enables seamless and precise workflow structures. Motivated from the aforementioned discussions, the paper presents a systematic review of the close interplay of SE and ML and possible interactions in different applications. Suitable research questions and case studies are presented for possible adoption scenarios that depict the close ML-SE interplay share with each other, with the concluding remarks. The paper forms useful insights for ML engineers, data science practitioners, and SE quality estimators towards the building of efficient and scalable software solutions.**

Index Terms*—Machine learning, Software Engineering, Effort estimation, Quality estimation

I. Introduction

The past few decades have been instrumental in the development of ML and SE-based practices in different applications. SE adopts systemic approaches towards development and ML provides systems with the ability to gather experience and learn without being explicitly programmed. In SE, numerous efforts are deployed in increasing accuracy and efficiency of code data, but improper versioning and heterogeneous workflows have been a major concern. In isolation, both technologies have been key drivers for the development of seamless applications. Fig. 1 shows the interplay of SE and ML-based on different parameters. As per the study reports by [1], the current SE market is estimated to reach 50 billion by 2030, growing at a compounded annual growth rate of 9 %. The details of the same is depicted in Fig. 1 (a). Thus, it is crucial to design effective strategies for workflow management in SE industries.

Through prospective automation of diverse applications around us, Artificial Intelligence (AI) and ML have fuelled the need for related software models. These components are predominantly data-driven and are expected to be dynamic, to adapt to changing Agile mindset requirements, and become a part of different applications [2]. Thus, ML models can be predominantly adopted to support SE operations in quality estimations (QE), quality assessment (QA), total quality management (TQM), mobile application development (MAD), and optimizing DevOps. As per the study by [1], the ML deployments in the defined SE operations would increase significantly by 2025. The details of the same is depicted in Fig. 1 (b). However, ML models have fundamental differences from traditional software models. These include non-monotonic erratic behavior due to entanglement, complex discovery, and management of data, different skills required for model reuse, and customization [3]. When used with for ML applications, these models have to be adjusted to the network domain [4].

For the same, SE operations or processes need to be carefully managed with different policy sets, and organizational structures. In agile methodology, we define scrum iterations with core-focused teams deploying complete software cycles. SE can help ML to build flexible agile models, and work-down structures, with proper data modeling for training models. Moreover, SE allows re-engineering the core modules with low cohesion and coupling strategies that allow easy testing and deployment strategies. With advent of security, decentralized SE models are also developed to integrate with ML. Also, with careful design strategies, legacy software codes could be made platform independent and run on heterogeneous machines. As per the study in [5], SE adoption in ML is measured based on the above tuning parameters. The details of the same is depicted in Fig. 1 (c). Thus, effort estimation is a crucial aspect with different ML models arise out based on the need for proper planning for efficient management of time, revenue, and resources. The cost of the project is predicted in dollars, efforts in person-months and the duration of the project is computed in calendar time.

Contrary to traditional systems, large software systems in a dynamic environment are a challenge today. There are several essential difficulties faced while developing software at such scale, including these given below:

- *Complexity*: A software consists of a large number of unique interacting parts which are encapsulated as different parts, and are invoked rather than replicated. It helps in hiding hard-to-find defects, thereby requiring significant additional work.
- *Invisibility*: While the changes made on a computer an inherently visible, changes to the software are not known

978-1-7281-7315-3/20/$31.00 ©2020 IEEE

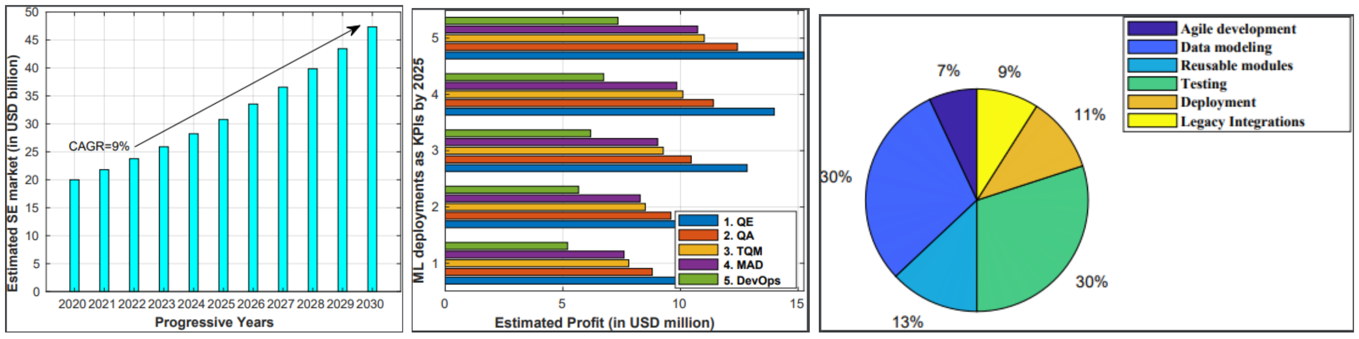

Fig. 1: Motivation towards an effective SE and ML interplay: (a) Estimated growth of SE market by 2030 [6], (b) ML interplay in SE based on quality and code parameters [1], (c) SE interplay in ML-based on tuning parameters [5].

.

to the user directly.

- *Conformity*: Software does not have to have underlying principles to which it must conform. However, it must be in terms of the representation of its parts and the environment it is working in.
- *Changeability*: Software systems must be changeable enough to facilitate higher chances of maintaining the value delivery over different phases of the life cycle.
- *Uniqueness*: One of the explicit properties of software systems, uniqueness has to be maintained throughout the model. This prevents the duplication of software systems, thus resulting in a higher productivity rate.

II. Related Work

As per Shah *et al.* [7], project estimation models can be broadly divided into two categories: Algorithmic models use estimation models for prediction. Some examples include Function Point (FP) Analysis, Lines of Code (LOC), Constructive Cost Model (COCOMO), etc [8]. Over the years, evolution has majorly been based on accurate estimations and the ability to accommodate general-purpose applications. Wen *et al.* [9] drew a systematic literature pertaining to effort estimation models for software development. By comparison, ML models outperform their counterparts on the basis of accuracy and estimation context. However, different ML models perform differently in place of different estimation contexts. Through a later study, Ali *et al.* [10] compared their study primarily to the former, including newer ML techniques and ensemble models. In Table I, we find the comparison of multiple approaches adopted, and their efficiency based on a defined set of parameters addressed.

III. Software Estimation Methodology

ML can be used in the field of software development to meet the needs of changing approaches, automation, adaptability, and scalability, and to keep pace with the increased developer productivity.

Before building quality evaluation models for software products, one first has to relate internal measurable artifacts like inheritance, and size to the required quality factors, one wants to assess [18], as shown in Fig. 2.

A. *Software Task formation*

Real-world software tasks can be solved using ML algorithms using a general set of guidelines, which are briefly described below.

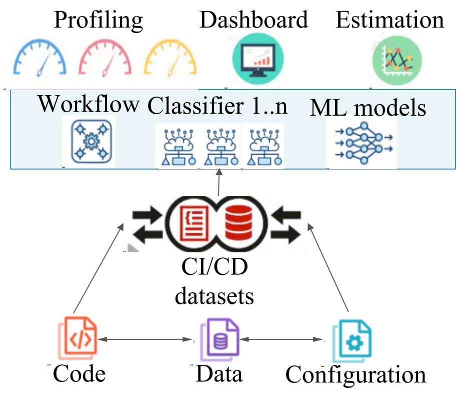

Fig. 2: Quality Assessment using Ml Pattern recognition [19]

1) Problem Formulation: The first task requires fitting a given real-world problem in conformity with the framework of a particular learning method chosen for the task. Several algorithms differ on bias, search strategies, requirements, and are based on different justifications for reasoning.

2) Problem Representation: Different representation formulations are used for many algorithms. Therefore, an appropriate representation of the knowledge and training data have to be prepared.

- *Data Collection*
  The collection of data depends on the learning process chosen from the lot. The chosen data set may have to be pre-processed before execution in the learning process.
- *Performing the learning process*
  One of the data and domain theory (if required) are obtained. Here, the data is divided into a training set and a test set, wherein the knowledge learned from the application of the learning process in the former is applied in the latter, and results are obtained, thus making this an iterative process [20].
- *Evaluation and Results* The process performed above is then assessed with a variety of evaluation parameters, the results of which are direct to the next step of work. If the results are inaccurate and unsatisfying, revisions may be needed. Once satisfactory results are obtained, the knowledge learned here is embedded in the required software development systems or software products.

TABLE I: A comparison of related approaches

SR No.	Author	Year	Objective	Cons	1	2	3	4	5	6	7
1	Shivhare *et al.* [11]	2014	Software estimation using different ML methods	Efficient algorithms stated but not discussed	✓	✗	✗	✓	✗	✗	✗
2	Prakash *et al.* [12]	2017	Survey on estimation models in traditional and agile methods	Absence of information on estimation by ML methods	✗	✓	✗	✗	✗	✓	✗
3	Meinke *et al.* [13]	2017	Survey of software applications benefitting from ML	Lack of enough information on crucial points addressed	✗	✓	✗	✗	✓	✗	✗
4	BaniMustafa *et al.* [14]	2018	Software effort estimation using ML techniques	Failure to use more data for training	✓	✗	✗	✓	✗	✓	✗
5	Ali *et al.* [10]	2019	Literature survey of software effort prediction using ML	Study solely focused on effort prediction	✓	✗	✓	✓	✓	✗	✗
6	Rahman *et al.* [15]	2019	Biometric authentication for Cloud Computation	Lower efficiency in preparation stage	✓	✓	✗	✓	✓	✓	✗
7	Rahman *et al.* [16]	2019	Case study on ML in SE	ML techques not discussed	✗	✓	✓	✗	✓	✓	✗
8	Shafiq *et al.* [17]	2020	Systematic mapping on ML for SE	Absence of proposal of efficient ML techniques	✗	✓	✓	✗	✓	✗	✗
9	Proposed Approach	2020	Interplay of ML & SE for quality estimations	-	✓	✓	✓	✓	✓	✓	✓

1. Different ML approaches 2. Various Software applications 3. Multiple data sets 4. Various Error Metrics 5. Research Questions 6. Alternate software estimation methods 7. SE in ML

B. *Research Questions*

Research questions are critical in the clarification and assessment of empirical evidence in software projects. These are specific to the study in question, and aid in covering all aspects. Upon analysis, we found the following questions used widely in software quality estimation.

- *RQ1*: Which data sets are used for estimation?
- *RQ2*: What are the criteria for the estimation of the process? (Evaluation parameters)
- *RQ3*: Does a combination of ML and non-ML techniques provide better results than individual means?
- *RQ4*: Which ML algorithms are used for software quality estimation?
- *RQ5*: Does the inclusion of expert judgment give better results, given the dynamic nature of software projects?
- *RQ6*: Do Analogy-based estimation methods outperform Algorithmic models (and vice versa)?
- *RQ7*: How do nature-inspired algorithms fare in terms of accuracy in comparison to traditional models?
- *RQ8*: DO ML algorithms result in better accuracy than non-ML algorithms?

C. *Evaluation Criteria*

Over the last decade, a numerous number of ML algorithms are used for prediction. All estimation models aim to increase the Percentage of Prediction (PRED) whilst decreasing the value of Mean Magnitude of Relative Error (MMRE) [7]. The mathematical equations of RE, MRE, MMRE and PRED are shown in equations 1, 2, 3, 4 and 5.

$$RE = (Estimated - Actual)/Actual \tag{1}$$

$$MRE = |Estimated - Actual|/Actual \tag{2}$$

$$MMRE = \sum MMRE/N \tag{3}$$

$$PRED(X) = A/N \tag{4}$$

$$EF = \frac{PRED(25)}{1 + MMRE} \tag{5}$$

Above, N is the number of projects at hand and A is the number of projects where MRE >= X. The value of X in equation 4 is usually considered to be 0.25 for effort estimation.

D. *Datasets Used*

Data-driven ML models require the use of historical project data for construction and evaluation. For the estimation of accuracy and other evaluation parameters, one has to take into account data sets on which the employed model is trained and validated. Ali *et al.* [10] and Wen *et al.* [9] have discussed some of the widely used data sets in their respective reviews. Among the many data sets used, we have the following used than others: Desharnias, COCOMO, ISBSG, Albrecht, Kemerer, NASA, Tukutuku, Maxwell, Finnish, and Miyazaki. As per Wen *et al.* [9], Desharnias was the most widely used data set, followed by COCOMO (2011), while later study by Ali *et al.* [10] showed the NASA data set as the most widely used data set, followed by COCOMO and ISBSG.

Adoption of SE in ML

With immense recent advancements in technology and machines, industries and companies are rapidly trying to build efficient ML models for various applications, and integrate them with their services, software, and functions. Traditional SE processes are not completely ideal and compatible with these ML applications.

AI-enabled systems are different than traditional systems in a lot of ways. Development tools, codes, algorithms, data, etc. vary a lot in these systems. Despite the prime focus on the accuracy of systems based on the ML algorithms, a lot of other operations, development, and SE tasks have to be performed in the background for proper building and maintenance of an application, which sometimes even accounts for technical debt [5]. Hence arises the need to develop more robust, complex and efficient development, and maintenance techniques and processes, to accommodate these more dynamic, data-driven systems.

IV. Case Studies and Evaluations

The section discusses three case studies highlighting the interplay of ML and SE in diverse domains.

A. *Case Study 1: Microsoft*

Fig. 3 depicts the general processes used by the Microsoft SE teams for the ML systems [6]. The teams have been working with Agile methods for ML-based SE processes. The ML workflow has been defined through various data-centered frameworks used for tasks such as data labeling,

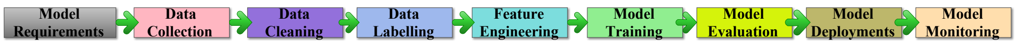

Fig. 3: Flow of a ML system in 9 stages [6].

feature engineering, model training, and model evaluation. A typical project team consists of employees with all ranges of expertise in ML tasks.

The case study was designed using two major approaches- interviews, and surveys. A diverse population of employees was chosen to gain broader viewpoints about all the related aspects. The analysis of the presented case study is depicted in Fig. 5 (a). The frequency of response sets of different employees is considered based on a survey challenge question set adopted from [6]. Based on the questions set, the difficulty is measured on measured SE parameters. The following practices were garnered from this:

- *End-to-end pipeline support*
  AI applications require a lot of computation power and the employees need to have an internal structure to carry out all the tasks hassle-free. As a result, strong pipelines, such as Azure ML IDE, have been developed to keep up with the constant loading of data.
- *Data availability, collection, cleaning, and management*
  The ML projects are always centered around data. Teams take to ensure that the data is available for use and reuse, and the new or manipulated data is properly versioned and managed. Hence, data management tools are often integrated with the ML frameworks to promote easier accessibility.
- *Education and Training*
  Continuous practices, such as conferences and discussions, are undertaken by the employees to aid everyone in learning more about the new techniques and practices in the world of ML and Data Science.
- *Model Debugging and Interpret-ability*
  Teams often try practicing better visualization of their code to improve the interpretation of code, and easier debugging.
- *Model Evolution, Evaluation, and Deployment*
  The ML software undergoes constant evaluation through automated test sets as well as human-oriented testing and error analysis. This whole process is often integrated into the whole software modeling rather than working on it separately, to allow faster deployment of the system.
- *Compliance*
  Everyone in the company must comply with the set of principles laid down by the company when practicing any SE or ML tasks.

Considering the above practices as challenges and the respondents being divided based on their experience with AI, the following results were identified through correlation and analysis:

- The most common problem amongst all the employees, regardless of their category, were data availability, collection, cleaning, and management.
- Employees with less experience identified Education and Training as their main challenge, while experienced employees felt that educating others was a major issue.
- As employees gain more experience, they tend to rely more on tools, and others.
- Employees with formal education understood the need to build pipelines and SRS documentation.

B. Case Study 2: SAP and Polytechnique Montreal

SAP and Polytechnique Montreal conducted a case study on a research project for analysis on SE and ML [21]. Their main aim was to identify various challenges in developing ML software, and suggest some proper guidelines and practices that practitioners should adhere to. The study's project was based on transactions, with the ML application being the automation of error detection and correction.

The SAP case study was conducted with a similar motive. The analysis is conducted based on Agile Sprint iterations and moving velocity of pipelined projects. The details of the study are presented in Fig. 5 (b). The study highlights that ML projects normally have more moving velocity due to simple engineered and automated modules, against conventional SE projects. For instance, after 70 sprint iterations, ML projects depict a velocity of 402.12 cost/sprint, whereas traditional SE project velocity is close to $221/24$ cost/sprints. The problems identified with non-ML projects are that manual corrections are required which result in a lot of time and high costs. A solution architecture based on an ML model is developed henceforth. An Agile methodology (SCRUM) [1] was used for the development of the application. Fig. 4 depicts various challenges faced in this project in different domains. Table II summarizes the recommendations for each challenge.

C. Case Study 3- Technical Debt Analysis by Google

Researchers at Google analyzed hidden technical debt in ML systems [5]. Technical debt is the cost or consequences of adopting Agile development methods, or basically, taking shortcuts to promote faster development [22]. For interpretation of the same, training losses of ML models are analyzed on debt patterns, based on kilo-lines of configured codes (KLOC). The details of the same are presented in Fig. 5 (c). It is observed that supervised models outperform non-supervised ML models, due to prior configuration knowledge. In supervised learning techniques, reinforcement strategy DeepQ learning is embedded and the ensemble has promising results, with low training loss. Similarly, combining supervised learning with deep-belief networks (DBN) results in the most accurate model with low losses. At 40 KLOC, supervised learning techniques have a loss percentage of 18.74 %, compared to 15.25 % in supervised and deep Q model, and the most accurate result is obtained with loss of 9.87 % in supervised and DBN approach. Thus, for technical debt patterns, deep networks form a suitable choice as per the study. Every system has some technical debts, but with ML systems, this debt is higher and hidden at times. Table III summarizes various types of debts that can be incurred and the primary reasons for their occurrence in ML systems.

Researchers and project teams should aim at reducing these technical debts as much as possible. A slight improvement in

TABLE II: Summary of the challenges identified in the project and their corresponding recommendations.

Challenge	Recommendation
Requirements Engineering	Feasibility analysis, solving conflicts in requirements, proper communication with users, keep an iterative check
Design	Modular design, flexibility, appropriate data handling, proper integration interface
Implementation	Use of frameworks and libraries, scalable, compatible, and portable
Integration	Functional integrity, and system design should be maintained
Testing	Rigorous and varied testing, proper debugging
Deployment	Platform difference to be considered, portable and scalable system, users should be unaffected
Problem Formulation	Full understanding of the problem, appropriate algorithm selection
Data Acquisition	Completeness, accuracy, consistency, and timeliness of data set, adaptable to changes, proper data requirements analysis
Data Preprocessing	Removal of noise, null values, etc., re-usability of data
Feature Extraction	Developing an automated pipeline, removal of noisy features, dimension reduction, regular review
Model Building	Appropriate algorithms, optimal solutions, balanced and quality data to be used, reuse existing solutions
Model Evaluation	Proper test cases, multiple evaluation parameters to be considered, evaluation of both the model and the system
Model Integration and Deployment	Functional integrity, platform difference to be considered, compatibility check, users should be unaffected
Model Management	Monitoring the model and the data, versioning of the system components, proper documentation
Ethics in AI Development	Preservation of user privacy and security, social gains to be prioritized over financial gains
Problem Understanding	Common understanding through varied perceptions
Focus on Objectives	Collaborative objectives, prioritization of objectives
Knowledge Transfer	Frequent interactions, proper identification of differences, industrial training
Professional Practice	Focus on objectives and not differences, proper interaction and collaboration
Data Privacy and Security	Sensitive user data should be private and secure, proper policies to be defined

TABLE III: Summary of hidden technical debts.

Type of Debt	Causes/Types
Abstraction boundaries	Entanglement of signals, correction cascades, undeclared consumers
Data Dependencies	Unstable Data Dependencies, Underutilized Data Dependencies, Static Analysis of Data Dependencies
Feedback Loops	Direct feedback loops, hidden feedback loops
ML-System Anti-Patterns	GLUE code, pipeline jungles, Dead Experimental Codepaths, abstraction debt, common smells
Configuration Debt	Potential for errors in configurations, messy configuration files
Other Debts	Data testing debt, reproducibility debt, process management debt, cultural debt

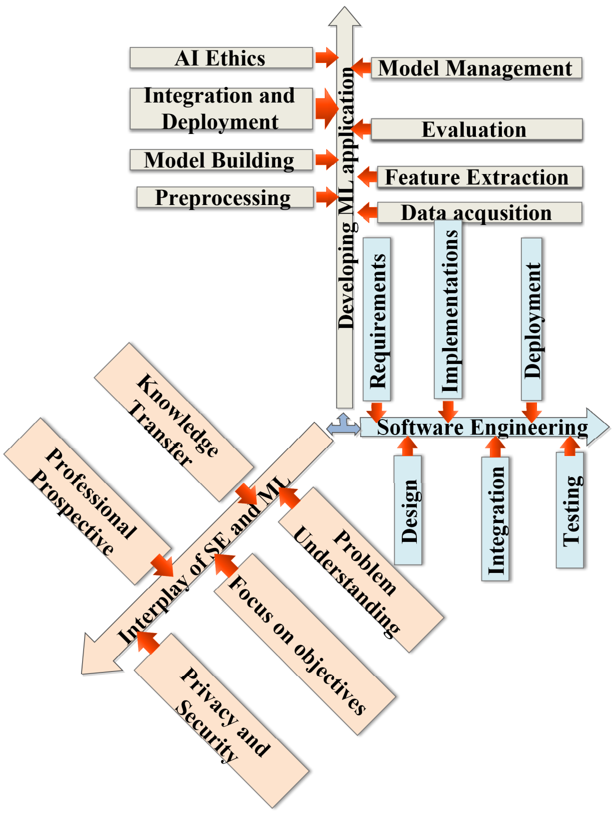

Fig. 4: Challenges in a ML project [21].

the accuracy of the ML model should not be undertaken at the cost of a huge increase in complexity and degradation of speed and other processes.

V. Open Issues and Challenges in Correlation between SE and ML

Although the above sections discuss in detail how SE and ML are related and dependant on each other, there must be a clearly defined boundary between the two domains. Neither all ML tasks are not SE tasks, nor all SE tasks are ML tasks. When working on projects strictly of one particular domain of the two, the task is relatively simple and comprehensible because the individual does not need to focus on complex processes, such as, integration with existing systems, changing the process itself, and optimizing the tasks. Code snippets and some code modules are enough to generate results with reasonable speed and accuracy. However, when people work on tasks combining the domains of SE and ML, it opens up endless possibilities.

On a general note, ML helps in automating and optimizing SE processes, whereas SE helps in developing proper ML-based systems with appropriate integration and maintenance. Recent advances in the IT industry have made both tasks inter-dependant. The real world and commercial purposes propose a lot of challenges and room for growth of the dependency of these domains, with the growth in ML-based software systems [23].

VI. Conclusion and Future Work

The paper discusses the emerging aspects of the interplay of SE and ML-based on effort and quality estimations. With the rise of flexible SE models, agility in life cycles, and increasing data-driven SE components, ML models could be integrated

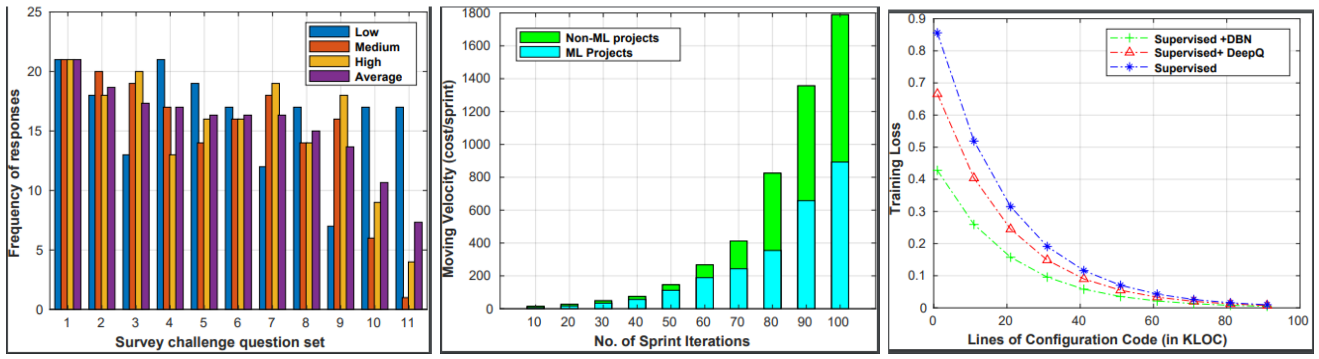

Fig. 5: Analysis of the presented case studies: (a) ML pipeline survey responses of Microsoft employees based on challenge question set [6], (b) Cost computation of Sprint iterations based on velocity models at SAP and Polytechnique Montreal [1], (c) Training losses for technical debt datasets against different schemes [5].

.

with SE processes to allow automation in software codes, testing, and validation strategies. Conversely, with ML models aging out, and increase in training data, the models tend to converge towards local maxima, and thus accuracy of models becomes monotonic. To handle the same, SE in ML allows re-engineering the data components, with flexible coupling strategies, that allows low-training losses with low MAE and precision losses. Thus, the interplay of SE and ML can scale up business automation, with logical and verifiable ecosystems. Thus, the paper presents the software estimation methodology, based on quality screening and research questions. We present the estimation parameters and available datasets, along with the flow engine of ML deployments in SE. In the future, the authors would explore the possibility of SE-ML fusion in terms of scaling-up operations, tool integration, and performance evaluation, through proposed models.

References

[1] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*. USA: Prentice Hall PTR, 1st ed., 2001.

[2] K. Chauhan, S. Jani, D. Thakkar, R. Dave, J. Bhatia, S. Tanwar, and M. S. Obaidat, "Automated machine learning: The new wave of machine learning," in *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, pp. 205–212, 2020.

[3] S. Tanwar, Q. Bhatia, P. Patel, A. Kumari, P. K. Singh, and W. Hong, "Machine learning adoption in blockchain-based smart applications: The challenges, and a way forward," *IEEE Access*, vol. 8, pp. 474–488, 2020.

[4] P. Bhattacharya, A. Singh, A. Kumar, A. K. Tiwari, and R. Srivastava, "Comparative study for proposed algorithm for all-optical network with negative acknowledgement (ao-nack)," in *Proceedings of the 7th International Conference on Computer and Communication Technology*, ICCCT-2017, (New York, NY, USA), p. 47–51, Association for Computing Machinery, 2017.

[5] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 2503–2511, Curran Associates, Inc., 2015.

[6] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann, "Software engineering for machine learning: A case study," in *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice*, ICSE-SEIP '19, p. 291–300, IEEE Press, 2019.

[7] M. A. Shah, D. N. A. Jawawi, M. A. Isa, M. Younas, A. Abdelmaboud, and F. Sholichin, "Ensembling artificial bee colony with analogy-based estimation to improve software development effort prediction," *IEEE Access*, vol. 8, pp. 58402–58415, 2020.

[8] R. Bhavsar, A. Thakkar, P. Sanghavi, and S. Tanwar, "Resolving conflicts in requirement engineering through agile software development: A comparative case study," in *International Conference on Innovative Computing and Communications* (S. Bhattacharyya, A. E. Hassanien, D. Gupta, A. Khanna, and I. Pan, eds.), (Singapore), pp. 349–357, Springer Singapore, 2019.

[9] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Information and Software Technology*, vol. 54, no. 1, pp. 41 – 59, 2012.

[10] A. Ali and C. Gravino, "A systematic literature review of software effort prediction using machine learning methods," *Journal of Software: Evolution and Process*, vol. 31, 07 2019.

[11] J. Shivhare and S. K. Rath, "Software effort estimation using machine learning techniques," in *Proceedings of the 7th India Software Engineering Conference*, ISEC '14, (New York, NY, USA), Association for Computing Machinery, 2014.

[12] B. Prakash and V. Viswanathan, "A survey on software estimation techniques in traditional and agile development models," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 7, pp. 867–876, 2017.

[13] K. Meinke and A. Bennaceur, "Machine learning for software engineering models, methods, and applications," 12 2017.

[14] A. Banimustafa, "Predicting software effort estimation using machine learning techniques," pp. 249–256, 07 2018.

[15] M. T. Rahman and M. M. Islam, "A comparison of machine learning algorithms to estimate effort in varying sized software," in *2019 IEEE Region 10 Symposium (TENSYMP)*, pp. 137–142, 2019.

[16] M. S. Rahman, E. Rivera, F. Khomh, Y.-G. Guéhéneuc, and B. Lehnert, "Machine learning software engineering in practice: An industrial case study," 2019.

[17] S. Shafiq, A. Mashkoor, C. Mayr-Dorn, and A. Egyed, "Machine learning for software engineering: A systematic mapping," 2020.

[18] H. Lounis, T. Gayed, and M. Boukadoum, "Machine-learning models for software quality: a compromise between performance and intelligibility," pp. 919–921, 11 2011.

[19] R. Rana and M. Staron, "Machine learning approach for quality assessment and prediction in large software organizations," 09 2015.

[20] R. Gupta, S. Tanwar, S. Tyagi, and N. Kumar, "Machine learning models for secure data analytics: A taxonomy and threat model," *Computer Communications*, vol. 153, pp. 406 – 440, 2020.

[21] M. S. Rahman, E. Rivera, F. Khomh, Y.-G. Guéhéneuc, and B. Lehnert, "Machine learning software engineering in practice: An industrial case study," 2019.

[22] Z. Codabux, B. Williams, G. Bradshaw, and M. Cantor, "An empirical assessment of technical debt practices in industry," *Journal of Software: Evolution and Process*, vol. 29, 08 2017.

[23] L. E. Lwakatare, A. Raj, J. Bosch, H. H. Olsson, and I. Crnkovic, "A taxonomy of software engineering challenges for machine learning systems: An empirical investigation," in *Agile Processes in Software Engineering and Extreme Programming* (P. Kruchten, S. Fraser, and F. Coallier, eds.), (Cham), pp. 227–243, Springer International Publishing, 2019.