SEYBOLD Report
ISSN: 1533 - 9211
DOI: 10.5281/zenodo.7157445

A STUDY OF BEST PRACTICES OF MACHINE LEARNING ALGORITHM IN SOFTWARE DEVELOPMENT PROCESS

[1]Dipali Brijpalsingh Tawar , [2]Dr. Deepika Pathak
[1]Research Scholar, Dept. of Computer Application, Dr. A.P.J Abdul Kalam University, Indore (M.P), India
[2]Associate Professor, Dept. of Computer Application, Dr. A.P.J Abdul Kalam University, Indore (M.P), India
[1]dipali.tawar@gmail.com, [2]deepikapathak23@gmail.com

ABSTRACT
The paper gives a data about software performance. The software performance is dissected and issues are investigated. Mathematical and engineering approaches and models for addressing performance issues are featured. The approaches to expand software performance are clarified. Significance of considering various factors here is accentuated. Performance of workers in undertakings is looked at and a calculation is produced for its enhancement. The created calculation will build the performance of the undertaking software. Investigations are led with the created calculation and the nearness of the performance of the undertakings is distinguished and delineated in the bar outline. A rundown of the best software to improve the performance is aggregated and broke down. The issue of software optimization is featured. Data on the necessities to the software optimization projects is given. The article likewise presents the approaches to build the performance of software engineers and propose strong suggestions here. A calculated model of software performance is created.

KEYWORDS: Software, Productivity, Optimization, Increase Productivity, Requirements.

INTRODUCTION
Software Engineering is a developing and arising field on the planet since software makes life more agreeable. Figure 1 shows software engineering originations. The significance of software is certain. In particular, in the here and now outline, the reality stays that PCs are vital in this day and age because of their widespread use in pretty much every field of life, particularly in trade, industry, medication, training, engineering, and agriculture. In the advanced time, data society is expanding quickly. PCs influence all cycles[1] in the public arena, including logical examination, economy, and for the most part change the manner in which individuals work and enter new zones of training. The investigation of new data innovations and their application in various regions lead to the creation and development of present day frameworks and programming dialects. The fundamental reason for mechanization is to save the civilization from latency and to liberate human from deadlines set for the execution of assignments. This is one of the primary patterns in the development of PC innovation. At the point when a software engineer has strong information on a programming language utilized for robotization issues, he/she can undoubtedly oversee on the other information. Also, a developer understands the construction of the program all the more effectively because of the information acquired. It ought to likewise be noticed that the investigation of software engineering, which has been a

SEYBOLD Report
ISSN: 1533 - 9211
DOI: 10.5281/zenodo.7157445

field of science for quite a while, is simple at any rate because of the way that programming permits the program to be applied to PCs for tackling explicit issues. Here, the software performance is one of the main points of interest. Software items are continually improving[2]: new highlights are added, UI changes, and so forth Software performance is a significant angle in building up any software item. Performance: capacity to deliver a specific number of items. All in all, it is a capacity to deliver a specific measure of item. To keep up the software extension measure, numerous compelling quality frameworks are created; which address an association's business requirements. Architects utilize different kinds of framework development measure model to coordinate the undertaking's life cycle. Different exercises might be done in different stages by a particular or team doing software development measure. Exemplary exercises acted in software development measure incorporates: System arranging, System requirements and advantages investigation, Project underwriting and project checking, System plan, Development, Software reconciliation and testing, System coordination and testing and documentation, Implementation and Maintenance[3]. The significant issue in software development is to control, how to instrument, utilizing certain abilities and inside specific limits. Software development lifecycle (SDLC) gives a short standpoint of how the unique critical thinking occasions might be done in different parts by a discrete or team doing software advance. The different SDLCs models are Waterfall, Iterative, Iterative and Incremental, Evolutionary Prototyping, Ad-hoc or Code-And-Fix SDLC.

Figure 1 Software Development Life Cycle

Numerous agile thoughts have been around since 70's or smooth previously and their substance is reported as a reaction against different traditional strategies. Agile strategies on a whole are new; they have solid roots throughout the entire existence of software engineering. The term agile can be characterized by quickness, correctness, and ease of quantity and it has likewise increased the public thought in late 1990's, agile methodologies were very much perceived to advance frameworks all the more rapidly with fractional time spent on examination and design. The quality of the software for the most part relies upon the software development life cycle (SDLC). The SDLC is a course utilized by software development industry to design, develop, and test high-quality software. The point of SDLC includes creating high-quality software that

SEYBOLD Report
ISSN: 1533 - 9211
DOI: 10.5281/zenodo.7157445
meets or surpasses client assumptions, arrives at fruition inside time, and cost assessment, and is straightforwardly identified with the client just as hierarchical fulfilment. It is a need for each association to adopt a minimal effort software development model. On the off chance that the minimal effort model can viably deliver high-quality software, it ought to be adopted to appreciate long haul benefits. It is essential for each association to look for high-quality and minimal effort software development models. Consequently, it is viewed as that a decent SDLC catches, checks, and carries out client requirements inside the time-box and purchased.

Existing Well-Known Models

The waterfall model is the principal, generally persuasive, and most ordinarily utilized process model. This model was suggested by Royce and remembers a linear or sequential execution of stages for a way with the end goal that the past stage gives input to the resulting stage, and this commonly follows the framework design corresponding to the main process model In order to defeat the critical limits of the waterfall model, an iterative model of software development was presented. In this methodology, requirements are gathered, and the task is developed and conveyed to the client through emphasess. A Rational Unified Process model (RUP) was presented with an equal working style wherein the new emphasis starts prior to delivering the current cycle, and this is extremely[4] time powerful. A twisting model is another illustration of the iterative model in development and from the conveyance perspective. In the twisting model, prototyping and design components are consolidated in a phase. Four major stages are associated with this model as follows: objective, danger, development and approval, and arranging. An extremely mainstream SDLC model that was named as the V-Model was developed in 1980. This model included an increased spotlight on testing to guarantee the quality of the software, and even each period of V-Model is related with testing. Extreme programming (XP) is the most ordinarily utilized strategy in agile procedure and includes the advanced form of the issues experienced in long development patterns of traditional development models. The XP method is portrayed by short development cycles, incremental masterminding, steady input, reliance on correspondence, and a transformative layout. The scrum procedure is an iterative and incremental process model to deliver or deal with any task. Basically, the scrum is a term that originates from system in rugby. It doesn't need or give a particular software development technique or practice that ought to be utilized by the scrum. The scrum just requires certain administration practices and tools in various periods of scrum to stay away from likely disarray because of flightiness and intricacy.

LITERATURE REVIEW

S. Velmurugan et al.[4], Quality is the main factor for software development as it fundamentally characterizes consumer loyalty that is straightforwardly identified with the accomplishment of a software project. The software process models is utilized to guarantee software quality, address an assortment of assignment settings, oversee project length, improve the process and reach to execute the process understanding, and to fitting understood guess for all undertaking settings. A few software processes models exist in software though with restricted degree. Given this perspective, this paper presents another software development life cycle model, "AZ-Model," for software development by presenting new exercises during

SEYBOLD Report
ISSN: 1533 - 9211
DOI: 10.5281/zenodo.7157445
software development life cycle. It defeats the impediments of traditional models and fundamentally impacts the creation of a quality item in a period box. This paper additionally presents a complete similar examination and factual investigations to inspect the meaning of AZ–Model for software development.

M. Niazi et al.,[5], In the course of the most recent decade, a ton of examination has been coordinated toward incorporating performance investigation into the software development process. Traditional software development strategies center around software correctness, presenting performance gives later in the development process. This methodology doesn't consider the way that performance issues may require extensive changes in design, for instance, at the software engineering level or far more detestable at the prerequisite investigation level. A few methodologies were proposed to address early software performance investigation. Albeit some of them have been effectively applied, we are still a long way from seeing performance examination incorporated into ordinary software development. In this paper, we present a complete survey of late exploration in the field of model-based performance expectation at software development time to evaluate the development of the field and point out promising examination bearings.

BEST PRACTICES OF MACHINE LEARNING

We categorized the challenges by card sorting interview and survey free response questions, and then used our own judgment as software engineering and AI researchers to highlight those that are essential to the practice of AI on software teams.

A. End-to-end pipeline support

As machine learning components have become more mature and integrated into larger software systems, our participants recognized the importance of integrating ML development support into the traditional software development infrastructure. They noted that having a seamless development experience covering (possibly) all the different stages described in Figure 1 was important to automation. However, achieving this level of integration can be challenging because of the different characteristics of ML[6] modules compared with traditional software components. For example, previous work in this field found that variation in the inherent uncertainty (and error) of data-driven learning algorithms and complex component entanglement caused by hidden feedback loops could impose substantial changes (even in specific stages) which were previously well understood in software engineering (e.g., specification, testing, debugging, to name a few). Nevertheless, due to the experimental and even more iterative nature of ML development[7], unifying and automating the day to-day workflow of software engineers reduces overhead and facilitate progress in the field.

B. Data availability, collection, cleaning, and management

Since many machine learning techniques are centered on learning from large datasets, the success of ML-centric projects often heavily depends on data availability, quality and management. Labeling datasets is costly and time-consuming, so it is important to make them available for use within the company (subject to compliance constraints). Our respondents confirm that it is important to “reuse the data as much as possible to reduce duplicated effort.” In addition to availability, our respondents[8] focus most heavily on supporting the following

DOI: 10.5281/zenodo.7157445

data attributes: “accessibility, accuracy, authoritativeness, freshness, latency, structuredness, ontological typing, connectedness, and semantic joinability.” Automation is a vital cross-cutting concern, enabling teams to more efficiently aggregate data, extract features, and synthesize labelled examples. The increased efficiency enables teams to “speed up experimentation and work with live data while they experiment with new models.”

C. Education and Training

The integration of machine learning continues to become more ubiquitous in customer-facing products, for example, machine learning components are now widely used in productivity software (e.g., email, word processing) and embedded devices (i.e., edge computing). Thus, engineers with traditional software engineering backgrounds need to learn how to work alongside of the ML specialists. A variety of players within Microsoft have found it incredibly valuable to scaffold their engineers’ education in a number of ways. First, the company hosts a twice-yearly internal conference on machine learning and data science, with at least one day devoted to introductions to the basics of technologies, algorithms, and best practices. In addition, employees give talks about internal tools and the engineering details behind novel projects and product features, and researchers present cutting-edge advances they have seen and contributed to academic conferences. Second, a number of Microsoft teams host weekly open forums on machine learning and deep learning[9], enabling practitioners to get together and learn more about AI. Finally, mailing lists and online forums with thousands of participants enable anyone to ask and answer technical and pragmatic questions about AI and machine learning, as well as frequently share recent results from academic conferences.

D. Model Debugging and Interpretability

Debugging activities for components that learn from data not only focus on programming bugs, but also focus on inherent issues that arise from model errors and uncertainty. Understanding when and how models fail to make accurate predictions is an active research area, which is attracting more attention as ML algorithms and optimization techniques become more complex. Several survey respondents and the larger Explainable AI community propose to use more interpretable models, or to develop visualization techniques that make black-box models more interpretable. For larger, multi-model systems[10], respondents apply modularization in conventional, layered, and tiered software architecture to simplify error analysis and debuggability.

E. Model Evolution, Evaluation, and Deployment

ML-centric software goes through frequent revisions initiated by model changes, parameter tuning, and data updates, the combination of which has a significant impact on system performance. A number of teams have found it important to employ rigorous and agile techniques to evaluate their experiments. They developed systematic processes by adopting combo-flighting techniques (i.e., flighting a combination of changes and updates), including multiple metrics in their experiment score cards, and performing human-driven evaluation for more sensitive data categories. One respondent’s team uses “score cards for the evaluation of flights and storing flight information: How long has it been flighted, metrics for the flight, etc.” Automating tests is as important in machine learning as it is in software engineering; teams

SEYBOLD Report
ISSN: 1533 - 9211
DOI: 10.5281/zenodo.7157445
create carefully put-together test sets that capture what their models should do. However, it is important that a human remains in the loop. One respondent said, “we spot check and have a human look at the errors to see why this particular category is not doing well, and then hypothesize to figure out problem source.” Fast-paced model iterations require more frequent deployment. To ensure that system deployment goes smoothly, several engineers recommend not only to automate the training and deployment pipeline, but also to integrate model building with the rest of the software, use common versioning repositories for both ML and non-ML codebases, and tightly couple the ML and non-ML development sprints and stand-ups.

F. Compliance

Microsoft issued a set of principles around uses of AI[10] in the open world. These include fairness, accountability, transparency, and ethics. All teams at Microsoft have been asked to align their engineering practices and the behaviors of fielded software and services in accordance with these principles. Respect for them is a high priority in software engineering and AI and ML processes and practices. A discussion of these concerns is beyond the scope of this paper.

G. Varied Perceptions

We found that as a number of product teams at Microsoft integrated machine learning components into their applications, their ability to do so effectively was mediated by the amount of prior experience with machine learning and data science. Some teams fielded data scientists and researchers with decades of experience, while others had to grow quickly, picking up their own experience and more-experienced team members on the way. Due to this heterogeneity, we expected that our survey respondents’ perceptions of the challenges their teams’ faced in practicing machine learning would vary accordingly. Two things are worth noticing. First, across the board, Data Availability[11], Collection, Cleaning, and Management, is ranked as the top challenge by many respondents, no matter their experience level. We find similarly consistent ranking for issues around the categories of end-to-end pipeline support and collaboration and working culture. Second, some of the challenges rise or fall in importance as the respondents’ experience with AI differs. For example, education and training is far more important to those with low experience levels in AI than those with more experience. In addition, respondents with low experience rank challenges with integrating AI into larger systems higher than those with medium or high experience. This means that as individuals (and their teams) gain experience building applications and platforms that integrate ML, their increasing skills help shrink the importance of some of the challenges they perceive. Challenges around tooling, scale[12], and model evolution, evaluation, and deployment are more important for engineers with a lot of experience with AI. This is very likely because these more experienced individuals are tasked with the more essentially difficult engineering tasks on their team; those with low experience are probably tasked to easier problems until they build up their experience.

EXPERIMENTAL METHODS

Software requirements are characterized during the beginning phases of a software development as a detail of what ought to be carried out. They are portrayals of how the

SEYBOLD Report
ISSN: 1533 - 9211
DOI: 10.5281/zenodo.7157445

framework ought to carry on, or of a framework property or quality. IEEE characterizes prerequisite investigation is a process of examining client needs to show up at a meaning of framework, equipment, or software requirements. Requirements investigation is important to the achievement of a frameworks or software project. The requirements ought to be archived, significant, quantifiable, testable, recognizable, identified with distinguished business needs and characterized to a degree of detail adequate for framework design. Software requirements incorporate business requirements, client requirements, framework requirements, outer interface prerequisite, useful requirements, and non-useful requirements[13]. In prerequisite articulations each individual business, client, practical, and non-useful necessity would show the characteristics. Attributes of prerequisite explanations are finished, correct, attainable, important, prioritized, unambiguous, and undeniable. Be that as it may, it's insufficient to have superb individual necessity explanations. So requirements assortments are utilized to gather a bunch of prerequisite or gathering of necessity.

Figure 2 Software performance Testing

Requirements should state what to do and the design ought to portray how it does this. Attributes of requirements are finished, steady, modifiable, and detectable which mirror the software quality. Quality is vigorously reliant on useful or non-practical requirements. In this segment we have dissected the different boundaries of necessity for example consistency, fulfilment or correctness and modifiable. Consistency implies giving unsurprising, viable and solid outcomes to the client. Consistency alludes to circumstances[14] where a particular contains no inside contradictions, while fulfilment alludes to circumstances where a detail involves all that is known to be "valid" in a specific setting. It contains all important information to stay away from uncertainty and need no intensification to empower appropriate execution and check. Correctness is generally intended to be the mix of consistency and fulfilment. Correctness is often more logically characterized as fulfilment of certain business objectives. Modifiable alludes to every prerequisite be extraordinarily named and communicated independently from others so you can allude to it unambiguously. In the event that its construction and style are changes to the necessity can be made effectively, totally and reliably.

SEYBOLD Report
ISSN: 1533 - 9211
DOI: 10.5281/zenodo.7157445

Using Machine Learning Algorithms for Software Development Performance

Machine learning manages the issue of how to construct PC programs that improve their performance at some assignment through experience. Machine learning algorithms have been used in data mining problems where enormous databases may contain significant certain consistencies that can be found consequently; poorly comprehended spaces where people probably won't have the information expected to develop successful algorithms; and areas where projects should powerfully adapt to evolving conditions. Learning an objective capacity from preparing data includes numerous issues (work portrayal, how and when to produce the capacity, with what given info, how to assess the performance of created work, and so forth). Major sorts of learning include: concept learning, decision trees, artificial neural networks, Bayesian belief networks, reinforcement learning, genetic algorithms and genetic programming, instance-based learning, inductive logic programming, and analytical learning. Table 1 sums up the primary properties of various sorts of learning. Decision Tree is a Supervised Machine Learning way to deal with tackle characterization and relapse problems by constantly parting data based on a specific boundary. The decisions are in the leaves and the data is part in the hubs. In Classification Tree[15] the decision variable is categorical (result as Yes/No) and in Regression tree the decision variable is ceaseless. Decision Tree has the accompanying advantages: it is appropriate for relapse just as arrangement issue, ease in translation, ease of handling categorical and quantitative qualities, fit for filling missing qualities in credits with the most plausible worth, high performance because of productivity of tree crossing algorithm. Decision Tree may experience the issue of over-fitting for which Random Forest is the arrangement which is based on gathering modeling approach.

Algorithm

INPUT: S, **where** S = *set of classified instances*
OUTPUT: *Decision Tree*
Require: $S \neq \varnothing$, $num\_attributes > 0$

```
1:  procedure BUILDTREE
2:      repeat
3:          maxGain ← 0
4:          splitA ← null
5:          e ← Entropy(Attributes)
6:          for all Attributes a in S do
7:              gain ← InformationGain(a, e)
8:              if gain > maxGain then
9:                  maxGain ← gain
10:                 splitA ← a
11:             end if
12:         end for
13:         Partition(S, splitA)
14:     until all partitions processed
15: end procedure
```

There are three ways to increase software performance:

- Using additional programs to increase software performance;

SEYBOLD Report
ISSN: 1533 - 9211
DOI: 10.5281/zenodo.7157445

- Using software capabilities to increase its performance;
- Increasing programmers' performance to increase software performance.

Software requirements are characterized during the beginning phases of a software development as a detail of what ought to be carried out. They are depictions of how the framework ought to carry on, or of a framework property or trait. IEEE characterizes prerequisite investigation is a process of considering client needs to show up at a meaning of framework, equipment, or software requirements. Requirements examination is important to the accomplishment of a frameworks or software project. The requirements ought to be archived, noteworthy, quantifiable, testable, recognizable, identified with distinguished business needs and characterized to a degree of detail adequate for framework design. Software requirements incorporate business requirements, client requirements, framework requirements, outer interface necessity, utilitarian requirements, and non-practical requirements. In prerequisite articulations each individual business, client, useful, and non-utilitarian necessity would display the characteristics. Qualities of prerequisite explanations are finished, correct, doable, essential, prioritized, unambiguous, and undeniable. Be that as it may, it's insufficient to have phenomenal individual prerequisite explanations. So requirements assortments are utilized to gather a bunch of prerequisite or gathering of necessity. Requirements should state what to do and the design ought to depict how it does this. Attributes of requirements are finished, predictable, modifiable, and recognizable which mirror the software quality. Quality is vigorously subject to utilitarian or non-useful requirements. In this part we have examined the different boundaries of prerequisite for example consistency, culmination or correctness and modifiable. Consistency implies giving unsurprising, viable and dependable outcomes to the client. Consistency alludes to circumstances where a detail contains no inside contradictions, though fulfilment alludes to circumstances where a determination involves all that is known to be "valid" in a specific setting. It contains all important information to maintain a strategic distance from equivocalness and need no intensification to empower appropriate execution and confirmation. Correctness is normally intended to be the mix of consistency and culmination. Correctness is often more even-minded characterized as fulfilment of certain business objectives.

Software Optimization Methods

Notwithstanding the steadily expanding viability of PCs, efficiency of human labour is rising intentionally, which is especially pertinent to profitability of software engineers. Condition can be improved through P-Modeling Framework, reversible semantic following and different philosophies that streamline the process of software development stages. For starting, the normal efficiency of labor of representatives of an organization is contrasted and the current profitability of labor of workers of different organizations. In the event that the indicators of the organization's performance surpass Fortune Global 500, this is an increase in performance. Efficiency doesn't just arrangement with flow and income. Labour is the central issue here. Labor is an outcome and advantage, as it is created by a worker. While assessing the matter of software developer, its environment is examined. It is difficult to recognize efficiency of labor in organization and its profitability in business-climate. The key issue facing a researcher is to

SEYBOLD Report
ISSN: 1533 - 9211
DOI: 10.5281/zenodo.7157445

increase performance. Criteria for optimization function are as follows: $y = \varphi(x1 , x2 ,..., xn) \rightarrow min$, thus, y - time of data processing; $x1 , x2 ,..., xn$ - all parameters (all affecting factors), thus, they can directly or indirectly affect performance; $xi \epsilon [ai , bi]$ - assigned field of the i-th factor.

The issue of low performance of information frameworks can be addressed by performing various assessments and changes of processes. Expanding the performance of existing frameworks may keep away from the acquisition of additional worker gear and save impressive assets to the spending plan. In such manner, the followings ought to be executed: investigating the framework; performance examination; performance review; performance engineering equipment optimization.

CONCLUSION

Software assumes a basic part in businesses, governments, and societies. To improve performance and quality of the software are important objectives of software engineering. A conversation on different augmentation models has been reachable in this paper. Albeit numerous development models exist, this paper examines various models out of those and the correlation incorporates the advantages and disadvantages of various models which can assist with choosing explicit model at explicit circumstances relying upon client demand and including business requirements. It additionally portray about agile technique, its different standards and steps. It likewise gives a correlation of agile and waterfall models, and additionally portrays the advantages of agile over traditional system. There are numerous restrictions and boundaries in different models. In future, our primary center is to lead a meeting from various industrialists, research researchers and figure the outcomes for assessment process. At present our survey identified with future work is done and we do computations utilizing chi-square methodology and plotting a model which will upsurge the performance of item and additionally figure cost and would be pertinent altogether sorts of software improvement process.

REFERENCES

[1]. N. M. A. Munassar and A. Govardhan, ‘‘A comparison between five models of software engineering,’’ Int. J. Computer. Sci. Issues, vol. 7, no. 5, pp. 95–101, 2010.

[2]. S. U. Khan, M. Niazi, and R. Ahmad, ‘‘Critical success factors for offshore software development outsourcing vendors: An empirical study,’’ in Proc. 11th Int. Conf. Product Focused Softw. Develop. Process Improvement (PROFES), Limerick, Ireland, 2010

[3]. S. Schneidera, R. Torkarb, and T. Gorschek, ‘‘Solutions in global software engineering: A systematic literature review,’’ Int. J. Inf. Manag., vol. 33, no. 1, pp. 119–132, 2013.

[4]. S. Velmurugan et al., ‘‘Software development life cycle model to build software applications with usability,’’ in Proc. Int. Conf. IEEE Adv. Computer., Commun. Inform. (ICACCI), Sep. 2014

[5]. M. Niazi et al., ‘‘Challenges of project management in global software development: A client-vendor analysis,’’ Inf. Softw. Technol., vol. 80, pp. 1–19, Dec. 2016.

DOI: 10.5281/zenodo.7157445

[6]. H. Lei et al., ‘‘A statistical analysis of the effects of scrum and Kanban on software development projects,’’ Robot. Computer.-Integr. Manuf., vol. 43, pp. 59–67, Feb. 2017.

[7]. C.-Y. Chen, P.-C. Chen, and Y.-E. Lu, ‘‘the coordination processes and dynamics within the inter-organizational context of contract-based outsourced engineering projects,’’ J. Eng. Technol. Manag., 2013.

[8]. P. Bedi, K. Upreti, A. S. Rajawat, R. N. Shaw and A. Ghosh, "Impact Analysis of Industry 4.0 on Realtime Smart Production Planning and Supply Chain Management," 2021 IEEE 4th International Conference on Computing, Power and Communication Technologies (GUCON), 2021, pp. 1-6, doi: 10.1109/GUCON50781.2021.9573563.

[9]. A. Tariq and A. A. Khan, ‘‘Framework supporting team and project activities in global software development (GSD),’’ in Proc. Int. Conf. Emerg. Technol. (ICET), 2012.

[10]. Bedi, P., Goyal, S.B., Rajawat, A.S., Shaw, R.N., Ghosh, A. (2022). A Framework for Personalizing Atypical Web Search Sessions with Concept-Based User Profiles Using Selective Machine Learning Techniques. In: Bianchini, M., Piuri, V., Das, S., Shaw, R.N. (eds) Advanced Computing and Intelligent Technologies. Lecture Notes in Networks and Systems, vol 218. Springer, Singapore. https://doi.org/10.1007/978-981-16-2164-2_23

[11]. B. Singh and S. P. Kannojia, “A model for software product quality prediction,” Journal of Software Engineering and Applications, Vol. 5, pp. 395-401, May 2012

[12]. Kumar, R., Singh, J.P., Srivastava, G. (2014). Altered Fingerprint Identification and Classification Using SP Detection and Fuzzy Classification. In: , et al. Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012), December 28-30, 2012. Advances in Intelligent Systems and Computing, vol 236. Springer, New Delhi. https://doi.org/10.1007/978-81-322-1602-5_139.

[13]. B. Singh and S. Gautam, “Hybrid Spiral Model to Improve Software Quality Using Knowledge Management,” International Journal of Performability Engineering, Vol. 12, Issue 4, pp. 341-352, July 2016.

[14]. Gite S.N, Dharmadhikari D.D, Ram Kumar,” Educational Decision Making Based On GIS” International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-1, Issue-1, April 2012.

[15]. Goyal, S.B., Bedi, P., Rajawat, A.S., Shaw, R.N., Ghosh, A. (2022). Multi-objective Fuzzy-Swarm Optimizer for Data Partitioning. In: Bianchini, M., Piuri, V., Das, S., Shaw, R.N. (eds) Advanced Computing and Intelligent Technologies. Lecture Notes in Networks and Systems, vol 218. Springer, Singapore. https://doi.org/10.1007/978-981-16-2164-2_25

[16]. S. Srivastava and R. Kumar, "Indirect method to measure software quality using CK-OO suite," 2013 International Conference on Intelligent Systems and Signal Processing (ISSP), 2013, pp. 47-51, doi: 10.1109/ISSP.2013.6526872.