Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 46 No. 2 (2025)

Comprehensive AI-Driven Agile Project Management and Resource Optimization Platform

M. I. M. Waseem, P. V. D. Sevindi, K. W. Jayinghe, Y. S. Padukka, P. K.W. Abeygunawardhana, K. T. S. Kasthuriarachchi

Dept. Of computer science and software engineering, Sri Lanka Institute of Information Technology, Malabe,Sri Lanka

Abstract: - Agile Software development is an iterative and flexible methodology that empowers teams to continuously adapt to changing requirements, collaborate closely, and deliver high quality software through rapid feedback loops. Traditional project management in Agile environments remains predominantly manual, leading to time consuming documentation, imprecise task breakdowns, reactive risk assessments, and inconsistent developer evaluations. This research introduces an AI driven project management platform that leverages Natural Language Processing, Machine Learning, and Predictive Analytics to automate critical workflows and enhance decision-making. The proposed system integrates four key components, automated documentation and summarization, an NLP-based speech-to-text module, abstractive summarization combined with sentiment analysis for generating structured project plans and meeting minutes, and an AI-based task extraction and prioritization engine that processes Software Requirement Specification documents using Named Entity Recognition, dependency parsing. And topic modelling. A hybrid predictive risk assessment module combining supervised and reinforcement learning approaches improves story point estimation and effectively predicts sprint risks. Developer evaluation is an automated system for analysing GitHub pull requests using machine learning-based metrices extraction and quality prediction, ensuring higher code standards and better collaboration. The platform utilizes data from open-source repositories and industry datasets. Integrating AI significantly enhances workflow efficiency, task accuracy, and early risk identification, ultimately improving team productivity and overall project success.

Keywords: *AI-driven project management, Natural Language Processing (NLP), Machine Learning, Agile Development, Task Prioritization, Developer Skill Assessment, Risk Prediction, Document Summarization, Story point estimation*.

1. Introduction

Agile software development has revolutionized the manage ment and delivery of modern software projects. It promotes an iterative and adaptive approach that emphasizes flexibility, collaboration, and continuous feedback to address evolving requirements [1]. Unlike traditional waterfall methodologies, Agile allows for the division of development cycles into smaller, more manageable sprints, which enables faster de livery of functional software [2]. This approach enhances efficiency, minimizes risks, and ensures that stakeholder ex pectations are met. Despite these advantages, Agile techniques present several challenges that can impede project efficiency and success, highlighting the need for advanced technical solutions [1].

One of the significant challenges in Agile environments is the process of recording and documenting meetings, which play a crucial role in teamwork and decision-making. Manual notetaking during meetings is not only time-consuming but also prone to inaccuracies. Important points and decisions may be missed when participants focus on note-taking in stead of engaging in the discussion. Studies have indicated that professionals, particularly managers, spend a substantial amount of time in meetings, and unproductive meetings can lead to significant financial losses, which negatively affect project outcomes. Software companies working on large scale projects often face employee turnover, requiring new team members to quickly familiarize themselves with the information produced by their predecessors. The traditional knowledge transfer process can take 2-3 months, during which new employees must review and understand all relevant docu mentation. This phase is not only time-consuming but also prone to misinterpretation and the risk of missing critical tasks, which can disrupt project continuity. Moreover, it often lacks consistency, which can lead to incomplete or inaccurate task breakdowns. Additionally, Manual code review processes are time-consuming and often prone to inconsistencies due to varying reviewer expertise, subjective judgments, and human errors [23]. In modern developer assessment, evaluating coding skills accurately becomes challenging, leading to potential biases and inefficiencies. Automating PR analysis can enhance consistency, reduce review time, and improve overall code quality assessment.

Inaccurate risk prediction is another persistent challenge in Agile development. Failing to identify risks early or assigning incorrect story points to tasks can create uncertainty within sprints [20]. These inaccuracies can lead to unforeseen obstacles, causing delays and disruptions to the project's timeline and budget. Without accurate estimations, teams often struggle to allocate work effectively, which can hinder project success [19].

To address these issues, this study presents a comprehensive AI-driven project management and resource optimization platform that integrates Natural Language Processing (NLP), Machine Learning (ML), and Predictive Analytics. The plat form comprises four core components.

Contextual Summarization Intelligent Document Generation Utilizes NLP and sentiment analysis to automate meeting transcription, extract meaningful insights, and generate structured reports, thereby reducing manual documentation efforts and enhancing decision-making in Agile environments.

AI-Based Task Creation Prioritization: Extracts tasks from SRS using NLP and machine learning to dynamically generate and prioritize tasks, establishing an automated workflow that improves task management efficiency and reduces the workload of project managers.

Intelligent Developer Skill Assessment: Analyses GitHub pull requests and evaluates developer skills based on code quality metrics such as complexity, cohesion, and maintain ability [25]. Employing machine learning models ensures objective, unbiased developer evaluations, thereby enhancing resource allocation within development teams.

Predictive Risk Analysis Sprint Management: Enhance story point estimation by integrating a Random Forest (RF) classifier with a Deep Q-Network (DQN) model, improving predictive accuracy through supervised and reinforcement learning. The model's predicted values are compared with the Scrum team's estimates to detect potential overestimations and underestimations. By analysing these discrepancies, the system effectively predicts sprint risks and facilitates proactive risk mitigation, leading to improved sprint outcomes.

By integrating AI-driven automation into Agile project management, this research aims to address critical limitations in manual processes, such as contextual summarization, in telligent document generation, task management, developer assessment, and risk prediction. Automating these tasks re duces manual effort, allowing teams to focus on high-priority activities. It also ensures objective role assignments and helps mitigate potential risks. Ultimately, this AI-powered approach enhances efficiency, productivity, and project outcomes, en abling Agile teams to adapt to dynamic project conditions.

2. Literature Review

Agile project management has used AI-driven automation to enhance efficiency, accuracy, and decision-making. Several studies have explored the automation of Agile environments. Our proposed system identifies existing limitations in con textual summarization for Agile documentation, automated task extraction from requirement specifications, evaluation of developer performance, and predictive risk identification in sprints. These advancements streamline project workflows, reduce manual effort, and improve collaboration among Agile teams, yet there remains a need for an integrated AI-powered framework that combines these components into a reliable system.

Modern automatic speech recognition (ASR) systems significantly enhance meeting documentation by accurately transcribing spoken content. OpenAI's Whisper, trained on 680,000 hours of multilingual data, offers robust transcription across various accents and domains without fine-tuning, achieving near human-level performance

[11]. This advancement addresses a critical challenge in meeting documentation by generating precise transcripts, reducing manual effort.

Since meeting transcripts are lengthy, abstractive summarization condenses them into concise and coherent summaries. Transformer-based models such as BART, T5, and PEGASUS have demonstrated strong performance in text summarization [13], [14], [15]. PEGASUS, designed for summarization tasks, achieves state-of-the-art results by learning to predict important sentences. These models can be fine-tuned using datasets like the AMI or ICSI meeting corpora to improve meeting summarization efficiency [10].

Sentiment analysis enhances meeting documentation by identifying emotional tones such as satisfaction, disagreement, or engagement. Rule-based models like VADER provide quick sentiment polarity detection, making them effective for short utterances [16]. However, transformer-based models like RoBERTa offer higher accuracy by capturing nuanced contextual sentiment [17]. RoBERTa fine-tunes pre-trained language models for sentiment classification, improving interpretation in meeting transcripts. Combining VADER's quick assessments with RoBERTa's deep contextual understanding ensures a comprehensive sentiment analysis, enabling better insights into team morale and discussion dynamics, ultimately supporting informed decision-making in meetings.

Software Requirements Specification (SRS) documents are vital for software development, containing both functional and non-functional requirements. However, translating these requirements into actionable tasks remains challenging. As Haris et al. note, the SRS document serves as the primary reference to ensure system functionality conformance [5].

Despite this importance, developers often measure product quality based only on released software, neglecting the original requirements. Traditional task extraction methods rely on manual processes, where developers read through documentation to identify tasks. This approach is time-consuming and prone to inconsistency, as different developers may extract different tasks from the same documentation. Additionally, the knowledge transfer period for new team members further exacerbates these inefficiencies. Recent research has explored automated task extraction. Atole et al. developed ”TASKNAVIGATOR,” a tool that uses natural language processing (NLP) to automatically extract task descriptions from software documentation [4]. However, much of the research on feature extraction has focused on source code rather than SRS documents. Extracting features from SRS documents has its advantages, as they represent the original intent for system functionality and serve as the basis for validation. Agile methodologies prioritize iterative development and task prioritization. Traditional methods, such as the MoSCoW technique, often rely on subjective assessments, failing to utilize available data or account for task interdependencies [9]. AI-driven approaches can consider multiple factors simultaneously, improving task prioritization and maximizing value delivery in each sprint.

Recent research has shifted towards machine learning models for software defect prediction and maintainability assessment. Studies in software quality metrics have shown that code complexity, cohesion, and coupling significantly impact software maintainability [24]. Furthermore, deep learning techniques like CodeBERT and Graph Neural Networks (GNNs) have enhanced automated analysis capabilities [26]. This paper expands on these foundations by proposing a machine learning-based approach for GitHub pull request (PR) quality assessment. Unlike existing approaches that focus solely on static analysis, our method integrates real-time API-driven analysis with predictive modelling, which enhances automation and consistency in PR evaluations. While previous studies have explored machine learning for defect prediction and maintainability, there is limited research on real-time, API driven PR quality evaluation within collaborative environments like GitHub. Most current solutions rely on offline batch processing, not integrating seamlessly into developers' work f lows. Additionally, past studies have not sufficiently addressed model interpretability or the contribution of various feature sets to PR quality predictions. Another significant gap is the lack of multi-language support, as most existing models are trained on a single programming language, limiting their generalizability [27]. This research aims to address these gaps by developing an end-to-end automated PR analysis system that incorporates real-time API processing, multi-language support, and enhanced model interpretability [28].

Identifying risks in a sprint is vital for Agile development, as accurate story point estimation enables teams to assess progress, prioritize user stories, plan iterations, and allocate resources effectively. Several studies have explored risk identification using different methodologies. R. Adel and H. Harb proposed a model using the Q-learning algorithm to assess risk factors like communication, coordination, and project management, structuring these risks into a Q-table for systematic learning [22]. Another study integrated Long Short-Term Memory (LSTM) and Recurrent Highway Networks to create a predictive model for story point estimation, improving accuracy in effort estimation [20]. Some researchers have used fuzzy logic tools to develop qualitative risk models, while quantitative risk assessments have been explored as well [19]. O. Kovalenko and O. Smirnov introduced a system that uses Failure Tree Analysis (FTA) for systematic risk identification [21]. Additionally, Monte Carlo simulations have been employed in Software Project Management (SPSM) models to predict project outcomes and risks [18]. The proposed predictive system identifies sprint risks by estimating story points with high accuracy. It uses a hybrid approach, combining a Random Forest (RF) classifier and a Deep Q-Network (DQN) model, integrating both supervised and reinforcement learning techniques. This model iteratively learns from past performance, refining its accuracy and adapting to changing project conditions, which is essential for maintaining timelines and improving sprint planning.

Existing research has validated the effectiveness of AI in Agile project management, but it typically focuses on individual functionalities rather than a unified AI framework. Most studies explore AI applications in isolation, without com bining them into a single AI-driven Agile project management system. This study seeks to address this gap by developing an integrated AI-driven project management framework that automates documentation, optimizes sprint workflows, eval uates developer performance, and enhances risk prediction, ultimately improving the overall effectiveness of Agile project execution.

3. Methodology

The research methodology is designed to ensure accuracy, reliability, and validity in the findings, with data collected from diverse sources, including real-time meeting recordings, Agile sprint reports, and software repositories. Speech-to-text models are employed to transcribe meeting discussions, while Agile project management tools like Jira and Trello offer structured data for analysis. The specific methodologies used for each key component of this research are outlined below.

A. Contextual Summarization and Intelligent Document Generation

Our pipeline transforms meeting audio into a structured document containing transcribed dialogue, identified speakers, an abstractive summary, and sentiment analysis. Speech recognition is performed using OpenAI's Whisper model, which converts spoken language into text while handling multiple languages and accents [11]. The transcript is then processed with Pyannote for speaker diarization, assigning labels to different utterances and improving transcript clarity in multi participant meetings [12].

To summarize long transcripts, we use transformer-based abstractive summarization models such as BART [13], T5 [14], and PEGASUS [15]. These models generate concise summaries capturing key decisions, action points, and discussions, with the best output selected using ROUGE scores.

Sentiment analysis is applied to understand the emotional tone of meetings. We employ a dual approach, using VADER for quick sentiment scoring and RoBERTa, a deep-learning model fine-tuned for sentiment classification, for nuanced emotional interpretation [16][17]. Aggregated sentiment scores generate sentiment trend charts, providing insights into emotional fluctuations during discussions.

The final output is a structured document integrating the transcript, summary, and sentiment analysis. Each speaker's contributions are timestamped, and visual elements such as sentiment trend graphs enhance usability. This AI-driven documentation system improves efficiency, reduces manual effort, and provides deeper insights into meeting discussions.

B. Extract tasks from SRS and prioritization

The preprocessing stage prepares the SRS document for analysis by converting it into a structured format and normalizing text. This includes

1) Document parsing to extract text content while preserving structural information
2) Sentence segmentation to divide text into individual sentences
3) Text normalization, including removal of stop words and special characters
4) Part-of-Speech (POS) tagging to identify the grammatical role of each word

Unlike prior methods relying on predefined requirement lists, our approach processes complete SRS documents directly, eliminating manual preparation and enhancing practicality.

To identify requirement sentences, we use a pattern-based approach with POS tagging. Requirement sentences in SRS documents often follow syntactic patterns. Following Haris et al., we identify common POS patterns like "The system shall [verb] [object]" or "The [actor] must be able to [verb] [object]," enabling automatic requirement identification [12].

Once identified, tasks are extracted using dependency parsing, which captures grammatical relationships. Our approach extends Atole et al. by incorporating additional features [11].

1) Identification of modal verbs (shall, must, should) that indicate requirements
2) Extraction of action verbs that indicate specific tasks
3) Identification of direct objects that represent the target of actions
4) Recognition of conditional clauses that specify constraints or conditions

Extracted tasks are structured with an action verb and associated objects.

We apply a multi-criteria prioritization algorithm tailored for agile environments, considering

1. Business value: The perceived value to end-users or customers
2. Technical risk: The potential for technical challenges or failures
3. Resource requirements: Estimated effort and expertise needed
4. Urgency: Time-sensitivity of the task

Each criterion is quantified using natural language analysis and machine learning [13]. Business value is assessed via requirement language, identifying core functionality, user experience, and business objectives. Technical risk is evaluated using complexity indicators [14].

A weighted scoring approach combines these factors, with machine learning determining weights based on project specific data, allowing adaptation to various contexts and team preferences [15].

C. Intelligent Developer Skill Assessment

GitHub PR data is retrieved using the GitHub API, focusing on programming languages such as Python, JavaScript, and C#. The dataset includes PR metadata, modified files, and historical review comments. Data preprocessing techniques such as deduplication, noise reduction, and irrelevant PR filtering are applied to ensure quality data. The system extracts key metrics from PR files, including, Cyclomatic Complexity (CCN),Measures logical branches in functions, A Random Forest classifier is trained to predict PR quality based on extracted features. The model is trained using labelled datasets from open-source repositories, categorized into 'Good' and 'Bad' PRs. Feature normalization and class balancing techniques, such as SMOTE (Synthetic Minority Over-sampling Technique) are applied during model training to improve prediction accuracy like Lines of Code (LOC), Counts source lines of code ,Function and Class Count: Identifies structural components, Code Cohesion and Coupling (CBO, RFC, LCOM): Evaluates maintainability and modularity, Variable and Literal Usage: Counts variable assignments, numeric literals, and function calls The system is built using FastAPI for backend processing and integrates with GitHub API for real-time PR analysis. The model is deployed as a REST API, providing endpoints for: User login and authentication, Fetching and analysing PRs, Generating reports on PR quality.

D. Predictive project analytics and risk assessment

The story point estimation process integrates a Random Forest (RF) classifier with a Deep Q-Network (DQN) model, combining supervised and reinforcement learning for improved accuracy. The training dataset consists of over 5000 user stories from open-source projects, each containing task titles, descriptions, and associated story points. Data preprocessing merges task titles and descriptions, replaces missing values with empty strings, and handles rare story point values to reduce sparsity. Label encoding standardizes values, and TF IDF vectorization with n-grams (1,2) converts text into numerical vectors. SMOTE addresses class imbalance by generating synthetic examples.

The RF classifier, configured with 500 estimators and balanced class weights, generates initial predictions by constructing multiple decision trees from random data subsets. These trees collectively vote on the final prediction, ensuring stability and accuracy while maintaining reproducibility through a fixed random state. The DQN model refines these predictions using reinforcement learning with two hidden layers of 256 neurons and ReLU activation to capture complex patterns. The output layer employs a linear activation function to produce Q values for adjustment actions. Hyperparameters include an exploration rate decaying from 1.0 to 0.01, a gamma factor of 0.95 for long-term reward optimization, and a learning rate of 0.001 for stable weight updates. An epsilon-greedy policy balances exploration and exploitation, while experience replay with a memory size of 10,000 stores past experiences to reduce data correlation. Mini-batch training accelerates convergence, and a target network updated periodically ensures training stability. The model learns from both successful and failed predictions, adjusting its weights to minimize prediction errors.

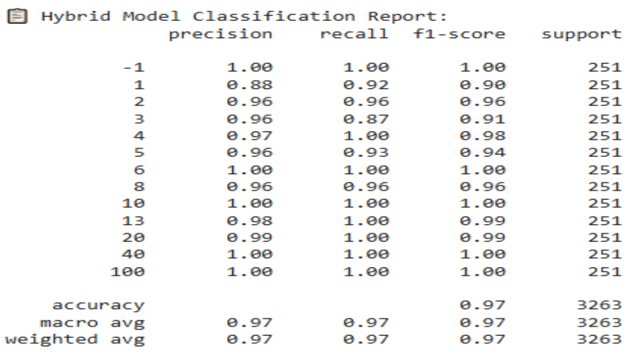
Once trained, the hybrid model's predictions are compared with Scrum team estimates to detect overestimations and un derestimations, identifying potential sprint risks. By leveraging both machine learning and reinforcement learning, the system dynamically adapts to project conditions, improving Agile project management and risk assessment.

To ensure the accuracy and generalizability of AI models, K-fold cross-validation is implemented, with datasets split into training and validation sets to prevent overfitting. Python libraries such as TensorFlow, Scikit-learn, and Pandas are used for model development, while visualization tools like Matplotlib and Seaborn help present results through charts and statistical evaluations. Performance metrics like precision, recall, and F1-score are used to evaluate AI effectiveness. Bias detection mechanisms, along with manual cross-validation, are incorporated to minimize errors and ensure the accuracy of AI-generated outputs.

4. Results and Discussion

AI-driven meeting documentation systems use sentiment analysis to classify discussions as positive, negative, or neutral, offering insights into engagement and discussion dynamics. By analysing utterances and sentiment trends over time, the system identifies emotional shifts indicating consensus, dis agreement, or key decisions. With an accuracy of 78.3%, the model captures emotional nuances, enhancing contextual understanding and ensuring that both factual content and underlying sentiments are recorded. This approach supports informed decision-making and improves meeting effectiveness.

The result of the hybrid model for estimating story points for user stories demonstrates a significant improvement in predictive accuracy. The training dataset consists of over 5000 user stories. The initial prediction model, utilizing a Random Forest (RF) classifier, achieved an accuracy of 83.54% after fine-tuning hyperparameters such as the number of trees and maximum depth to prevent overfitting. The Deep Q-Network (DQN) model further optimized hyperparameters, including the learning rate, discount factor, and batch size, to improve

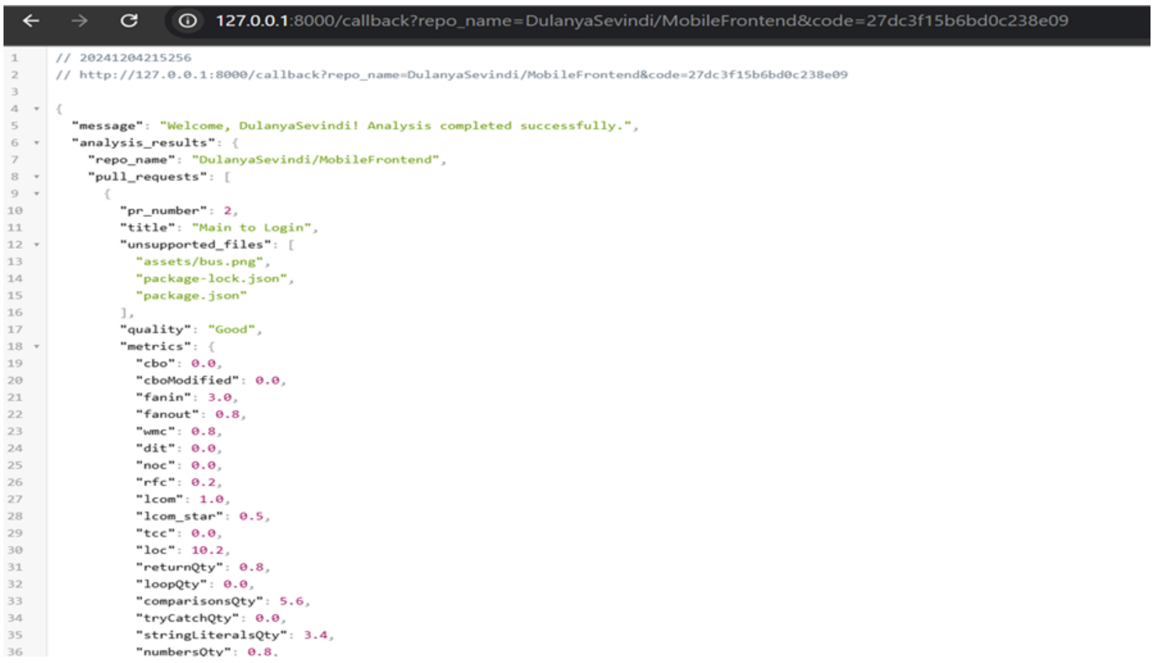


```
Hybrid Model Classification Report:
              precision    recall  f1-score   support

          -1       1.00      1.00      1.00       251
           1       0.88      0.92      0.90       251
           2       0.96      0.96      0.96       251
           3       0.96      0.87      0.91       251
           4       0.97      1.00      0.98       251
           5       0.96      0.93      0.94       251
           6       1.00      1.00      1.00       251
           8       0.96      0.96      0.96       251
          10       1.00      1.00      1.00       251
          13       0.98      1.00      0.99       251
          20       0.99      1.00      0.99       251
          40       1.00      1.00      1.00       251
         100       1.00      1.00      1.00       251

    accuracy                           0.97      3263
   macro avg       0.97      0.97      0.97      3263
weighted avg       0.97      0.97      0.97      3263
```

Fig. 1. Accuracy of hybrid model

convergence and performance. By combining the RF classifier with the DQN model, the hybrid model achieved an accuracy of 97.24%, demonstrating how the hybrid approach significantly enhances estimation accuracy compared to the standalone models.

The dataset used for training and testing consists of 10,000 GitHub pull requests (PRs) from various open source projects, encompassing programming languages such as Python, JavaScript, and C. For model training, 80% of the data is utilized, while the remaining 20% is reserved for testing the model's performance. The evaluation metrics for the model show promising results, with a precision of 88.2%, recall of 85.7%, and an F1-score of 86.9%, indicating a well-balanced trade-off between identifying relevant PRs and minimizing false positives. These metrics suggest that the model performs effectively in predicting PR quality.



```
// 20241204215256
// http://127.0.0.1:8000/callback?repo_name=DulanyaSevindi/MobileFrontend&code=27dc3f15b6bd0c238e09

{
  "message": "Welcome, DulanyaSevindi! Analysis completed successfully.",
  "analysis_results": {
    "repo_name": "DulanyaSevindi/MobileFrontend",
    "pull_requests": [
      {
        "pr_number": 2,
        "title": "Main to Login",
        "unsupported_files": [
          "assets/bus.png",
          "package-lock.json",
          "package.json"
        ],
        "quality": "Good",
        "metrics": {
          "cbo": 0.0,
          "cboModified": 0.0,
          "fanin": 3.0,
          "fanout": 0.8,
          "wmc": 0.8,
          "dit": 0.0,
          "noc": 0.0,
          "rfc": 0.2,
          "lcom": 1.0,
          "lcom_star": 0.5,
          "tcc": 0.0,
          "loc": 10.2,
          "returnQty": 0.8,
          "loopQty": 0.0,
          "comparisonsQty": 5.6,
          "tryCatchQty": 0.0,
          "stringLiteralsQty": 3.4,
          "numbersQty": 0.8,
```

Figure 1 Fig. 2. Developer Skill Assessment

5. Future Works

Future improvements for the proposed system involve several key enhancements. Real-time transcription will be improved for more accurate meeting discussions, and speaker recognition will be refined to better identify team members. Sentiment analysis will be enhanced to capture more nuanced emotions, and user studies will help refine the system's usability. AI-driven task extraction and prioritization will be integrated with popular project management tools, version control systems, and CI/CD pipelines to streamline workflows. Additionally, developer skill assessment models, such as Gradient Boosting, SVM, and Deep Learning, will be explored for improved accuracy through feature engineering, hyperparameter tuning, and real-time updates. Risk assessment will be strengthened by integrating DevOps tools, allowing for more precise sprint predictions through the inclusion of deployment data and testing results in story point estimation

6. Conclusion

This research introduces a comprehensive AI-driven project management and resource optimization platform that integrates Natural Language Processing (NLP), Machine Learning (ML), and Predictive Analytics. The platform improves efficiency by reducing human input and enhancing adaptability to chang ing project requirements. By integrating automation into key processes, the system ensures more manageable sprints, ac celerates software delivery, and minimizes risks. Automated document generation, optimized task prioritization, developer skill assessment, and project risk prediction contribute to better project planning and execution. This allows teams to focus more on innovation and problem-solving, ultimately driving overall project success and meeting stakeholder expectations.

References

[1] H. Fawareh, Y. Al-Smadi, R. Saadeh, F. A. Fawareh, A. Elrashidi and H. M. Al-Shdaifat, ”A Comparative Study between Agile and Waterfall Methodologies during Software Development Process,” 2024 25th International Arab Conference on Information Technology (ACIT), Zarqa, Jordan, 2024, pp. 1-5, doi: 10.1109/ACIT62805.2024.10877239.

[2] P. Jain, A. Sharma and L. Ahuja, ”The Impact of Agile Software Development Process on the Quality of Software Product,” 2018 7th International Conference on Reliability, Infocom Technologies and Op timization (Trends and Future Directions) (ICRITO), Noida, India, 2018, pp. 812-815, doi: 10.1109/ICRITO.2018.8748529.

[3] S. Sadaf, S. Iqbal, A. Saba and M. KamarMohsin, ”An extended adaptive process model for agile software development methodology,” 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), Kerala, India, 2017, pp. 1373-1378, doi: 10.1109/ICICICT1.2017.8342770.

[4] R.N Atole, B. Jadhav Komal, U. Gaikwad Aaditi, and R. Lagad Ankita, “Automatic Development of Task Extraction and Task Recommendation from Software Documents” International Journal of Innovative Science and Research Technology, vol. 3, no. 2, February 2018.

[5] M. Syauqi Haris, S. Syauqi, and M. S. Budiman, ”Automated Features Extraction from Software Requirements Specifications (SRS) Docu ments as The Basis of Software Product Line (SPL) Engineering” International Journal of Information Technology and Computer Science, vol. 5, no. 3, pp. 279-292, December 2020.

[6] Y. Bugayenko, M. Farina, A. Kruglov, W. Pedrycz, Y. Plaksin, G. Succi, “Automatically Prioritizing Tasks in Software Development,” 2016.

[7] G.T Tsvetanka, “Applying Data Classification for Predicting the Task Priorities of Software Projects” 23rd International Symposium INFOTEH-JAHORINA, 20-22 March 2024.

[8] H. Alaidaros, A. Bakodah, S.F. Bamsaoud, “A Review on Requirements Prioritization Approaches of Software Project Management” nternational Conference on Intelligent Technology, System and Service for Internet of Everything (ITSS-IoE), 2022.

[9] K.S. Ahmad, N. Ahmad, H. Tahir, S. Khan, “A Fuzzy based MoSCoW Method for the Prioritization of Software Requirements” International Conference on Intelligent Computing, Instrumentation and Control Tech nologies (ICICICT), 2017.

[10] V. Rennard, G. Shang, J. Hunter, and M. Vazirgiannis, “Abstractive Meeting Summarization: A Survey,” Transactions of the Association for Computational Linguistics, vol. 11, pp. 861–884, 2023.

[11] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust Speech Recognition via Large-Scale Weak Supervi sion,” arXiv preprint arXiv:2212.04356, 2022.

[12] H. Bredin et al., “pyannote.audio: neural building blocks for speaker diarization,” in Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP), 2020.

[13] M. Lewis et al., “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension,” in Proc. 58th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 7871–7880, 2020.

[14] C. Raffel et al., “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer,” J. Mach. Learn. Res., vol. 21, no. 140, pp. 1–67, 2020.

[15] J. Zhang, Y. Zhao, M. Saleh, and P. Liu, “PEGASUS: Pre-training with Extracted Gap-Sentences for Abstractive Summarization,” in Proc. 37th Int. Conf. Machine Learning (ICML), vol. 119, pp. 11328–11339, 2020.

[16] C. J. Hutto and E. E. Gilbert, “VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text,” in Proc. 8th Int. AAAI Conf. Weblogs and Social Media (ICWSM), Ann Arbor, MI, 2014, pp. 216–225.

[17] Y. Liu et al., “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” arXiv preprint arXiv:1907.11692, 2019.

[18] M. I. Lunesu, R. Tonelli, L. Marchesi, and M. Marchesi, ”Assessing the Risk of Software Development in Agile Methodologies Using Simulation,” IEEE Access, vol. 9, pp. 134240-134256, Oct. 2021, doi: 10.1109/ACCESS.2021.3115941.

[19] V. Anes, A. Abreu and R. Santos, ”A New Risk Assessment Approach for Agile Projects,” 2020 International Young Engineers Forum (YEF ECE), Costa da Caparica, Portugal, 2020.

[20] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies, ”A Deep Learning Model for Estimating Story Points,” IEEE Transactions on Software Engineering, 1 July 2019.

[21] O. Kovalenko, O. Smirnov, A. Kovalenko, and S. Kavun, ”Quantitative Risk Assessment Method Development in the Context of the SDLC model,” 2021 IEEE 8th International Conference on Problems of Info communications, Science and Technology (PIC ST), Kharkiv, Ukraine, 2021.

[22] R. Adel, H. Harb, and A. Elshenawy, ”A Multi-agent Reinforcement Learning Risk Management Model for Distributed Agile Software Projects,” 2021 IEEE Tenth International Conference on Intelligent Computing and Information Systems (ICICIS), pp. 512-520, 2021, doi: 10.1109/ICICIS52592.2021.9694252.

[23] J. Romano et al., ”Code Metrics for Software Quality Evaluation,” IEEE Software, vol. 35, no. 2, pp. 74-81, 2018.

[24] J. Lin et al., ”Machine Learning-Based Static Analysis of Software Defects,” in Proc. IEEE ICSE, 2020, pp. 453-462.

[25] T. Yamada, K. Yokoyama, and S. Nakae, ”Periodic Developer Metrics in Software Defect Prediction,” 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), Melbourne, Australia, May 2023.

[26] S. Gupta, B. Gupta, and S. Gupta, ”A Novel Method for Technical Candidate Assessment using Github Repository Inspection Automation,” 2022 International Conference on Computer Communication and Infor matics (ICCCI), Coimbatore, India, 2022.

[27] G. R. Bergersen, D. I. K. Sj.berg, and T. Dyba, ”Construction and Validation of an Instrument for Measuring Programming Skill,” IEEE Transactions on Software Engineering, vol. 40, no. 12, pp. 1163-1184, Dec. 2014.

[28] T. McCabe, ”A Complexity Measure,” IEEE Transactions on Software Engineering, vol. SE-2, no. 4, pp. 308-320, 1976.