Real-Time Feedback for BDD Test Scenarios Using AI-Based Classification

David Brandão
Department of Computer Engineering
University of Pernambuco
Recife, Pernambuco, Brazil
d.brandao@sidi.org.br

Denis Marques
Department of Computer Engineering
University of Pernambuco
Recife, Brazil
denis.marques@upe.br

Cleyton Rodrigues
Department of Computer Engineering
University of Pernambuco
Recife, Brazil
cleyton.rodrigues@upe.br

Wylliams Santos
Department of Computer Engineering
University of Pernambuco
Recife, Brazil
wbs@upe.br

ABSTRACT

Behavior-Driven Development (BDD) has gained widespread adoption as a means to align software behavior with stakeholder expectations, yet maintaining high-quality scenarios remains challenging at scale. Manual review of Gherkin-based steps is often slow, inconsistent, and prone to oversight, leading to structural errors, semantic inconsistencies, and reduced maintainability. To address these issues, this work proposes a hybrid automated analysis framework that combines Natural Language Processing (NLP) and Machine Learning (ML) to improve both the clarity and correctness of BDD artifacts. The framework consists of two complementary components: a rule-based validator that inspects linguistic and structural adherence to established BDD conventions and a supervised classifier that assigns each step to one of three semantic categories: Precondition, Action, or Expected Result regardless of its original Gherkin keyword. Models were trained on a balanced synthetic dataset of 1,500 labeled steps and validated against a large-scale industrial repository from a leading global manufacturer of laptops and mobile devices, ensuring external validity. Performance was measured using macro-averaged accuracy, precision, recall, and F1-score, alongside statistical significance testing to compare algorithms. The best results were achieved by Support Vector Machines and gradient boosting models, which outperformed neural and transformer-based approaches. Designed for near real time operation, the framework can be applied to any Gherkin compatible library and any supported natural language, enabling broad applicability across projects. It integrates seamlessly into development workflows, including pull requests and CI/CD pipelines, to provide continuous, automated feedback on BDD scenarios. Findings suggest that hybrid NLP–ML solutions are effective in scaling quality assurance for agile both Test and DevOps teams, while reducing the manual effort required for review and maintenance.

KEYWORDS

Behavior-Driven Development, Gherkin, Natural Language Processing, Machine Learning, Hybrid Approach, Test Automation, Software Testing, Step Classification, Rule-based Validation, Continuous Integration, Continuous Delivery

1 Introduction

The Software Development Life Cycle (SDLC) is a structured process that guides software creation from initial conception to deployment and maintenance, comprising stages such as Planning, Requirements Analysis, Design, Development, Testing, Deployment, and Maintenance [2]. Among these, the testing phase is critical for verifying and validating that the software meets its specified requirements and is free from defects.

Within modern development practices, BDD has emerged as a methodology that not only facilitates automated testing but also serves as a living documentation medium. Introduced by Dan North [19], BDD promotes collaboration between developers, testers, and business stakeholders through a shared, domain-specific language [21]. Its emphasis on clarity and continuous feedback aligns closely with agile principles [11] and has become increasingly relevant as AI-based approaches begin to influence software testing practices [23].

At the heart of BDD lies the Gherkin language, developed by Aslak Hellesøy as part of the Cucumber project [14]. Gherkin enables stakeholders to define system behavior in an accessible, structured format, typically in plain English or other supported languages, using three key constructs: 'Given' (preconditions), 'When' (actions), and 'Then' (expected results) [5, 14]. These constructs form the backbone of test scenarios, ensuring clarity, consistency, and alignment between technical and business perspectives [5].

Although BDD helps ensure that requirements are clearly specified and testable, its effectiveness is highly dependent on adherence to best practices, such as consistent use of third-person descriptions, logical sequencing of steps, and avoidance of redundancy. In practice, these conventions are not always followed, especially by less experienced practitioners, which can compromise maintainability and introduce defects [21].

The literature reveals no existing approach that unifies best-practice validation and semantic step classification for BDD scenarios. Related works, such as the RClassify model for rule categorization in requirements documents [2], address similar challenges in other domains. However, the specific combination of rule-based validation and ML classification in the context of Gherkin remains unexplored. Given the proven success of NLP and ML in tasks involving the extraction and classification of natural language rules

[1, 2, 17, 22], this study investigates whether such techniques can enhance the identification of structural and semantic patterns in BDD test scenarios, thereby improving their quality and maintainability.

Building on this motivation, the primary objective of this research is to design and implement a hybrid classification system that leverages NLP and ML techniques to automatically analyze and classify the BDD test steps written in Gherkin. The proposed system performs both semantic validation, ensuring adherence to structural conventions, and functional categorization into conditions, actions, and expected results, providing a comprehensive mechanism to improve the clarity, correctness, and maintainability of BDD artifacts.

2 Background

The adoption of advanced practices in software development has reinforced the role of methodologies such as BDD, designed to bridge the gap between business needs and technical implementation. In BDD, requirements are collaboratively specified by developers, testers, and stakeholders in a shared, domain-specific language [8]. This approach supports the description of expected system behavior in structured and standardized scenarios, improving requirement comprehension and enabling the creation of automated tests aligned with business objectives [21].

2.1 Software Testing

Software testing is a fundamental component of software quality, shaped by principles from quality management pioneers such as Juran [16] and Deming [6], whose work emphasized planning, control, continuous improvement, and defect prevention. Over time, testing evolved from a debugging-focused activity into a structured, evaluation-oriented discipline integrated throughout the development lifecycle [12], with milestones such as the V-model for aligning verification and validation [10] and agile practices like TDD [3].

Conceptually, testing involves designing test cases, executing them against the system under test, and evaluating the results to determine conformity with expected behavior. It may be performed manually or by automation. Manual testing remains essential in exploratory and usability contexts, but automation is increasingly favored for its efficiency, repeatability, and scalability [7]. Automated testing not only accelerates execution but also facilitates regression coverage, enabling teams to maintain quality as systems evolve [13].

The testing process itself can be organized into layers, as described in ISO/IEC/IEEE 29119-2 [15], which include organizational-level processes (e.g., policies and strategies), test management processes, and dynamic testing processes. Automation can support all these layers, not only in test execution, but also in test case generation, data management, monitoring, statistical analysis, and reporting, providing both operational efficiency and improved traceability.

In practice, effective test automation contributes to cumulative regression coverage, allowing repeated execution of a growing suite of automated cases over time. This improves productivity, enhances defect detection, and reduces long-term costs. By strategically selecting which tests to automate and integrating them into continuous integration and delivery pipelines, organizations can align testing with modern quality management principles, ensuring that systems meet requirements with consistency and speed [9].

2.2 Automated Requirements Analysis and Classification

As modern testing increasingly incorporates automation to improve efficiency, consistency, and scalability, new opportunities emerge for applying AI and NLP to analyze not only code and execution results, but also textual artifacts that define system behavior. In the case of BDD, these artifacts written in structured natural language - can be automatically inspected and classified, enabling the enforcement of best practices and the detection of semantic or structural issues before execution.

ML, a subfield of AI, enables systems to learn patterns from data without explicit rule programming. When combined with NLP, which focuses on enabling computers to interpret and process human language, it becomes possible to automate the analysis of complex textual artifacts, including BDD scenarios [8, 25].

NLP has been extensively applied to the extraction of business rules in various industries, such as data integration between heterogeneous systems in the energy sector [27], classification and spelling checks for low-resource languages [18], and the extraction of metadata from unstructured scientific documents using a combination of NLP and Computer Vision (CV) techniques [4].

In the context of business rule classification, ML and Deep Learning approaches have been employed to categorize rules in requirements documentation. Notable works include deep learning for business rule classification in software requirement specifications [1], ML techniques to improve requirements elicitation [17], and DocToModel, which automates the extraction of structured information from unstructured text [22]. These studies demonstrate the effectiveness of NLP and ML for textual analysis, with some combining both paradigms [28].

However, most of these solutions focus solely on classification or rule extraction. Few address the combined use of NLP and ML in BDD testing, particularly for enforcing best practices and improving scenario quality. In practice, ensuring semantic correctness and stylistic adherence, such as using third-person narrative, maintaining logical sequencing, and avoiding redundancy, remains a manual and error-prone task.

3 Methodology

The proposed methodology integrates NLP and ML techniques to enhance both the quality and the correctness of the BDD test steps written in Gherkin. It is structured around two independent yet complementary modules: (i) a rule-based validator that performs static and semantic analysis to enforce adherence to established BDD conventions, and (ii) a supervised ML classifier that categorizes steps into semantic roles, Precondition, Action or Expected Result, independent of their original Gherkin keywords.

This section details the system architecture, the implementation of each module, the research design, and the datasets and metrics used for evaluation.

3.1 Architecture Overview

The proposed framework consists of two core modules, each addressing a distinct aspect of BDD step analysis.

The first, the **NLP Rule Validator**, applies static and semantic checks to verify compliance with established Gherkin best practices. While it does not perform classification, it generates targeted feedback on structural and linguistic deviations, such as incorrect sequencing, redundant conjunctions, or violations of third-person narrative. This proactive validation improves the readability, consistency, and maintainability of scenarios before they are subjected to classification.

The second, the **ML Classifier**, assigns each Gherkin step to one of three semantic roles: Precondition, Action, or Expected Result independent of its original keywords. This functional classification enables the detection of logical misplacements and reinforces scenario coherence by focusing on the intended meaning of the step rather than its keyword.

The outputs of these two modules are integrated into a unified feedback pipeline: the NLP module flags deviations from writing conventions, while the ML module provides semantic categorization. Together, they support automated quality control, facilitate real-time review, and enhance the long-term maintainability of BDD test suites.

3.2 Hybrid NLP–ML Architecture and Dataset

The proposed framework adopts a hybrid design that integrates two distinct but complementary modules: an NLP-based rule validator and an ML-based step classifier. Each addresses a different dimension of the BDD scenario analysis the NLP component enforces structural and linguistic best practices, while the ML component performs semantic role classification. Together, they form a cohesive architecture capable of improving both the quality and correctness of Gherkin-based test steps (Figure 1).

NLP Rule Validator. This module applies heuristic rules derived from established Gherkin and BDD best practices [20, 24, 26] to perform static and semantic analysis on each step. It identifies violations that may compromise clarity, maintainability, or adherence to conventions, such as the use of imperative verbs instead of third-person descriptions, excessive conjunctions, or incorrect step sequencing. The implemented rules include:

- **Third-person validation:** Ensures that steps are written in a neutral, descriptive voice.
- **Conjunction redundancy:** Detects repeated use of conjunctions such as `And` or `But` in a single step.
- **Step sequence check:** Validates logical ordering (e.g., `Given` precedes `When` and `When` precedes `Then`).
- **Excessive repetition:** Flags steps starting with the same keyword or conjunction unnecessarily.
- **Semantic similarity:** Detects redundant or overly similar phrases across steps in the same feature file.

Operating as a static analysis stage, the NLP validator is independent from the ML training pipeline but ensures that scenarios enter classification in compliance with BDD standards.

ML Classifier. This component assigns each `Gherkin` step to one of three semantic roles: *Precondition*, *Action*, or *Expected Result*, independent of the original keyword (`Given`, `When`, `Then`). Steps are preprocessed and vectorized using a hybrid representation combining TF-IDF with Word2Vec embeddings, capturing both term-level importance and semantic relationships. The following supervised learning algorithms were evaluated:

- **Support Vector Machine (SVM)**
- **Light Gradient Boosting Machine (LightGBM)**
- **Extreme Gradient Boosting (XGBoost)**
- **Multilayer Perceptron (MLP)**
- **Logistic Regression (LogReg)**
- ***k*-Nearest Neighbors (kNN)**
- **Bidirectional Encoder Representations from Transformers (BERT, fine-tuned)**
- **Gaussian Naive Bayes (GNB)**
- **Random Forest (RF)**
- **Extra Trees Classifier (ExtraTrees)**
- **CatBoost (CB)**
- **Long Short-Term Memory (LSTM)**

Models were trained on a balanced synthetic dataset of 1,500 manually annotated steps and evaluated on a large-scale real-world industrial dataset, in addition to 10-fold cross-validation on the synthetic set. Performance was measured using the macroaveraged accuracy, precision, recall, and F1 score.

Dataset. Due to confidentiality constraints, real project data could not be used for training. Instead, a synthetic dataset of 1,500 manually labeled `Gherkin` steps was created, equally distributed across the three target categories (500 per class). This data set was carefully designed to replicate the structure and linguistic patterns of real scenarios without omitting proprietary information. For external validation and generalization assessment, access was granted to a large-scale industrial repository from a global technology company specializing in laptops and mobile devices. This real-world dataset served exclusively for final evaluation, ensuring that the reported results reflect practical applicability in production environments.

Integration. In the final workflow, the NLP validator flags structural and semantic issues, while the ML classifier provides semantic categorization. This sequential process enables the detection of both writing inconsistencies and logical misplacements, producing comprehensive automated feedback for improving the clarity, correctness, and maintainability of scenarios.

3.3 Type of Research

This work is classified as *applied research* because it addresses a concrete and practical problem in software testing: enhancing the quality of scenarios written in `Gherkin` through automated analysis.

The research has a *descriptive* objective, aiming to investigate how NLP and ML techniques can be integrated to perform both qualitative validation and functional classification of test steps. In this context, the NLP module enforces established best practices through rule-based checks, while the ML module assigns each step to one of three semantic categories: Precondition, Action, or Expected Result.

From a methodological point of view, the approach is *mixed method*. The NLP component delivers qualitative, nonpredictive

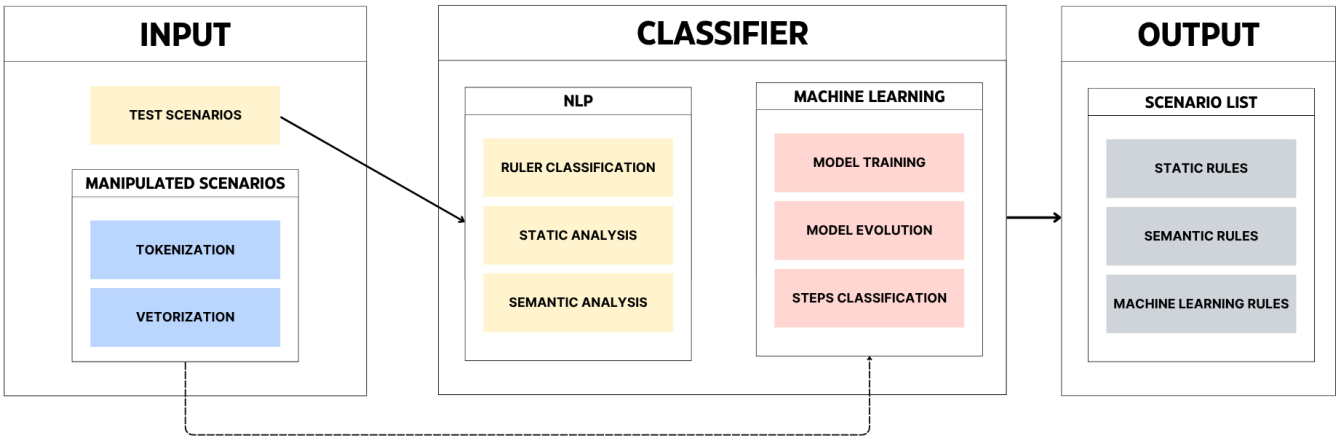

Figure 1: Hybrid NLP–ML architecture and data flow for BDD step validation and classification

analysis of structural and semantic adherence, whereas the ML component is quantitatively evaluated using macroaveraged Accuracy, Precision, Recall, and F1-score metrics. This dual perspective enables a comprehensive assessment that captures both the structural integrity and semantic correctness of test scenarios.

3.4 Tools and Techniques

The implementation of this study was developed in Python, combining NLP and ML techniques to improve the quality of scenarios written in `Gherkin`. The solution integrates open-source libraries for text processing, vectorization, and model training.

For text processing, **SpaCy** was employed for tokenization and semantic analysis, providing efficient tools to validate linguistic structure. Rule validation was implemented through custom logic designed to enforce best practices, such as maintaining third-person narrative and ensuring the correct sequencing of *Given*, *When*, and *Then* steps.

Text vectorization adopted a hybrid approach combining **TF-IDF** and **Word2Vec**. TF-IDF quantifies the importance of a term within a document, calculated as:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

where:

$$\text{TF}(t, d) = \frac{\text{Count of term } t \text{ in document } d}{\text{Total number of terms in } d}, \quad \text{IDF}(t) = \log\left(\frac{N}{\text{DF}(t)}\right)$$

Word2Vec generates dense vector representations that capture semantic relationships between words. Combining both methods enables the model to leverage syntactic importance and semantic similarity simultaneously, improving classification accuracy.

The models were implemented using **Scikit-learn** (Random Forest, Naive Bayes, XGBoost) and **TensorFlow/Keras** (LSTM, BERT), enabling the integration of traditional algorithms and deep learning architectures into the hybrid validation–classification framework.

3.5 Evaluation Metrics

The performance of the proposed ML models was quantitatively evaluated using standard multiclass metrics: Accuracy, precision, recall, F1 score, and standard deviation. These are widely used in classification tasks and provide a comprehensive view of predictive capability and model stability.

Accuracy measures the proportion of correct predictions over the total number of predictions:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision evaluates the proportion of correctly predicted positive instances among all instances predicted as positive:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall measures the proportion of actual positive instances correctly identified:

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-score is the harmonic mean of Precision and Recall, balancing the two:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

For Precision, Recall, and F1 score, *macro averaging* was applied to ensure that each class contributes equally, regardless of its frequency in the data set.

Standard Deviation was computed to quantify variability in performance across cross-validation folds, providing insights into model robustness.

These metrics form the basis for the comparative analysis presented in Section 4.

4 Results

The proposed hybrid NLP–ML framework was evaluated using both the synthetic and real-world datasets described in Section 3.2. All models were cross-validated 10-fold in the synthetic dataset, followed by a final evaluation in the real-world dataset. Performance was assessed using the metrics defined in Section 3.5.

The evaluation aimed to:

(1) Identify which supervised learning model achieved the highest and most consistent performance in classifying Gherkin steps into the three semantic roles.
(2) Assess whether performance on the synthetic dataset generalizes to real-world BDD test scenarios.

Macro-averaged values were used to ensure balanced evaluation across all target classes. Table 1 (presented later) summarizes the results obtained from experiments on the real dataset, while the key comparative insights are outlined below.

4.1 Model Performance Comparison

Table 1 presents the macroaveraged performance of each model in the hold-out test set. Considering all metrics, **SVM** achieved the highest overall performance, with the best accuracy and precision while maintaining balanced recall and F1-score. **LightGBM** followed closely, with slightly lower precision but the best F1-score, indicating strong generalization capability. **XGBoost** ranked third, delivering consistent results across all metrics, followed by **MLP**, which also showed competitive outcomes. In contrast, **LSTM** and **CatBoost** obtained substantially lower scores, suggesting poor adaptation to the dataset and the need for further tuning or alternative architectures.

Table 1: Performance metrics of classification models (macro-averaged), ordered by accuracy.

Model	Accuracy	Precision	Recall	F1-score
SVM	**0.770**	**0.812**	**0.770**	0.765
LightGBM	0.769	0.794	0.769	**0.770**
XGB	0.727	0.750	0.726	0.727
MLP	0.705	0.706	0.709	0.700
LogReg	0.652	0.664	0.655	0.647
kNN	0.572	0.573	0.574	0.569
BERT	0.548	0.742	0.557	0.484
GNB	0.527	0.591	0.531	0.522
RF	0.500	0.341	0.509	0.407
ExtraTrees	0.491	0.335	0.499	0.400
CB	0.334	0.111	0.333	0.167
LSTM	0.322	0.107	0.333	0.162

4.2 Performance Analysis

The results in Table 1 reveal notable trends in model behavior. While deep learning architectures such as LSTM and BERT are often expected to outperform traditional methods, in this task, simpler models—particularly **SVM**, **LightGBM**, and **XGB**—delivered superior and more consistent results.

SVM achieved the highest overall performance, with the best accuracy (0.770) and precision (0.812), while maintaining balanced recall and F1-score. **LightGBM** closely followed, with slightly lower precision but the best F1-score (0.770), indicating strong generalization capability. **XGB** ranked third, delivering consistent results across all metrics, followed by **MLP**, which also showed competitive outcomes.

In contrast, **LSTM** and **CB** exhibited the lowest scores, likely due to the limited dataset size and a mismatch between their architectural assumptions and the available features. Although **BERT** obtained a high precision (0.742), its lower recall (0.557) and F1 score (0.484) suggest conservative predictions: fewer positive classifications but higher certainty.

4.3 Statistical Significance Analysis

To verify whether the observed differences are statistically significant, the **Friedman test** was applied to the F1 scores of all models:

$$\text{Friedman statistic} = 54.5692, \quad p\text{-value} = 9.29 \times 10^{-8}$$

Since $p < 0.05$, the null hypothesis that all models perform equally is rejected, confirming statistically significant differences among them.

4.4 Pairwise Comparisons Using Student's T-Test

To provide deeper insight into the performance differences highlighted in Table 1, pairwise two-tailed **Student's T-tests** were conducted on the F1-scores of selected models representing: (i) the top-performing group and (ii) an extreme performance gap scenario.

Table 2: Pairwise T-test results for selected model comparisons (F1-scores).

Model Comparison	t-statistic	p-value
SVM × LightGBM	-1.4226	0.19460
SVM × LSTM	46.6135	< 0.0001

The SVM × LightGBM comparison yielded $p = 0.19460$, indicating no statistically significant difference between the two best-performing models, suggesting both are equally strong candidates. The SVM × LSTM test ($p < 0.0001$) confirmed an extreme disparity, with LSTM achieving results close to random classification. These results reinforce the robustness of ensemble and margin-based methods when combined with hybrid TF-IDF and Word2Vec features, while also underscoring the importance of aligning model architectures with the characteristics of the target dataset.

Interpretation of individual results.

- **SVM vs LightGBM**: Both models showed very similar performance across folds, with variations not large enough to be statistically significant. This suggests that either could be used interchangeably in production without noticeable impact on classification quality.
- **SVM vs LSTM**: The large positive t-statistic and extremely low p-value confirm a clear and consistent superiority of SVM over LSTM, with the latter performing only marginally better than random guessing for this classification problem.

5 Conclusion

This study presented a hybrid approach for evaluating and classifying Behavior-Driven Development (BDD) test scenarios written in

Gherkin, integrating Natural Language Processing (NLP) and Machine Learning (ML) techniques. The proposed solution addresses two critical challenges in BDD-based test automation: (i) ensuring compliance with syntactic and semantic best practices and (ii) accurately classifying test steps according to their functional roles.

The architecture combines a rule-based NLP validator, responsible for static analysis to enforce structural and stylistic conventions, with a supervised ML classifier that categorizes steps as *Preconditions*, *Actions*, or *Expected Results*, independently of the original Gherkin keywords. This dual-layered design provides both qualitative feedback on scenario quality and quantitative classification outputs, forming a comprehensive framework for automated BDD step analysis.

A distinctive aspect of this evaluation is that the models were trained exclusively on a synthetically generated dataset and tested on a real-world BDD repository provided by a global technology company specializing in laptops and mobile devices. This industrial dataset enabled a realistic and high-impact assessment of model performance, reinforcing both the external validity and practical applicability of the approach. The results in Table 1 show that **SVM** achieved the highest accuracy (0.770) and precision (0.812), while maintaining strong recall (0.770) and F1-score (0.765). **LightGBM** followed closely, with the highest F1-score (0.770) and balanced performance across all metrics. **XGBoost** ranked third, showing consistent results across all measures, while **MLP** also delivered competitive performance with metrics around 0.70.

In contrast, neural architectures such as **LSTM** and transformer-based models like **BERT** did not surpass the traditional algorithms that perform the best. Although BERT achieved high precision (0.742), its recall (0.557) and F1-score (0.484) were notably lower, indicating a tendency toward conservative predictions. Models such as **CatBoost** and **LSTM** performed poorly, with results approaching those of random classification.

The Friedman test confirmed that the observed differences among models were statistically significant ($\chi^2 = 54.5692$, $p = 9.29 \times 10^{-8}$). Furthermore, a pairwise Student's T-test between XGBoost and MLP yielded $t = 5.0624$, $p = 0.00134$, indicating a significant performance advantage for XGBoost in terms of F1-score. These findings highlight both the robustness of ensemble learning methods, particularly gradient boost, and the effectiveness of hybrid feature engineering strategies that combine TF-IDF and Word2Vec for structured textual classification tasks.

In conclusion, the proposed hybrid NLP–ML framework not only enforces BDD best practices, but also achieves strong classification performance on industrial data in the real world, even when trained exclusively on synthetic datasets. This demonstrates the viability of synthetic data generation for training models in domains with limited labeled resources. In addition, the system's capability to deliver near real-time step-level feedback makes it highly suitable for integration into CI/CD pipelines, enabling continuous quality control of BDD scenarios during development and prior to deployment. In future work, we plan to extend the NLP validator with autocorrection features, integrate semantic role labeling for deeper contextual analysis, and explore transfer learning approaches to further enhance cross-domain generalization.

ACKNOWLEDGEMENTS

The authors acknowledge the support of the SiDi Institute for Research and Development, whose collaboration was essential to the successful completion of this work.

REFERENCES

[1] P. R. Anish, P. Lawhatre, R. Chatterjee, V. Joshi, and S. Ghaisas. 2022. Automated labeling and classification of business rules from software requirement specifications. In *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice*. 53–54.
[2] R. Asha, N. Padmalata, P. Ajim, K. Piyush, and K. Vinay. 2023. RClassify: Combining NLP and ML to Classify Rules. In *2023 IEEE 31st International Requirements Engineering Conference (RE)*.
[3] Kent Beck. 2003. *Test Driven Development: By Example.* Addison-Wesley.
[4] Zeyd Boukhers and Azeddine Bouabdallah. 2022. Vision and Natural Language for Metadata Extraction from Scientific PDF Documents: A Multimodal Approach. *2022 ACM/IEEE Joint Conference on Digital Libraries (JCDL)* (2022).
[5] Adwait Chandorkar, Nitish Patkar, Andrea Di Sorbo, and Oscar Nierstrasz. 2022. An Exploratory Study on the Usage of Gherkin Features in Open-Source Projects. *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)* (2022).
[6] W. Edwards Deming. 1993. *The New Economics.* MIT Press.
[7] Elfriede Dustin, Jeff Rashka, and John Paul. 1999. *Automated Software Testing: Introduction, Management, and Performance.* Addison-Wesley.
[8] Muhammad Shoaib Farooq, Uzma Omer, Amna Ramzan, Mansoor Ahmad Rasheed, and Zabihullah Atal. 2023. Behavior Driven Development: A Systematic Literature Review. *IEEE Access* (2023).
[9] Mark Fewster and Dorothy Graham. 1999. *Software Test Automation: Effective Use of Test Execution Tools.* Addison-Wesley.
[10] Kevin Forsberg and Harold Mooz. 1991. The Relationship of System Engineering to the Project Cycle. In *Proceedings of the National Council on System Engineering (NCOSE)*. 57–65.
[11] Martin Fowler. 1999. *Refactoring: Improving the Design of Existing Code.* Addison-Wesley Professional.
[12] David Gelperin and William C. Hetzel. 1988. The Growth of Software Testing. *Commun. ACM* 31, 6 (1988), 687–695.
[13] Mary Jean Harrold. 2000. Testing: A Roadmap. In *Proceedings of the Conference on the Future of Software Engineering*. 61–72.
[14] Aslak Hellesøy, Matt Wynne, and Steve Tooke. 2017. *The Cucumber Book: Behaviour-Driven Development for Testers and Developers.* Pragmatic Bookshelf.
[15] ISO/IEC/IEEE. 2013. ISO/IEC/IEEE 29119-2:2013 - Software and Systems Engineering – Software Testing – Part 2: Test Processes. Accessed on: September 4, 2024.
[16] Joseph M. Juran. 1989. *The Quality Trilogy: A Universal Approach to Managing for Quality.* Juran Institute.
[17] Mohammed Lafi and Akram abdelQader. 2023. Automated Business Rules Classification Using Machine Learning to Enhance Software Requirements Elicitation. *2023 International Conference on Information Technology (ICIT)* (2023).
[18] Diellza Nagavci Mati, Mentor Hamiti, Besnik Selimi, and Jaumin Ajdari. 2021. Building Spell-Check Dictionary for LowResource Language by Comparing Word Usage. *2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)* (2021).
[19] Dan North. 2006. Introducing BDD. *Better Software* 8, 2 (2006), 29–34.
[20] Dan North. 2006. Introducing BDD. *Better Software* (2006).
[21] OneDayTesting. 2019. Gherkin: Concepts and Benefits. https://blog.onedaytesting.com.br/gherkin/ Accessed on: September 4, 2024.
[22] Asha Rajbhoj, Padmalata Nistala, Vinay Kulkarni, Shivani Soni, and Ajim Pathan. 2023. DocToModel: Automated Authoring of Models from Diverse Requirements Specification Documents. *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (2023).
[23] Stuart Russell and Peter Norvig. 2020. *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.
[24] John Ferguson Smart. 2014. *BDD in Action: Behavior-driven development for the whole software lifecycle.* Manning Publications.
[25] Khushboo Taneja and Jyoti Vashishtha. 2022. Comparison of Transfer Learning and Traditional Machine Learning Approach for Text Classification. *2022 9th International Conference on Computing for Sustainable Global Development* (2022).
[26] Matt Wynne and Aslak Hellesoy. 2017. *The Cucumber Book: Behaviour-Driven Development for Testers and Developers.* Pragmatic Bookshelf.
[27] Jiakai Xiao, Wei Du, Zhengxiang Xu, and Yang Qian. 2023. Cross-system Data Integration Based on Rule-based NLP and Node2Vec. *2023 8th International Conference on Data Science in Cyberspace (DSC)* (2023).
[28] Y. Ye, X. Xie, H. Jin, and D. Wang. 2021. A Hybrid Model Combined with SVM and CNN for Community Content Classification. In *2021 IEEE 23rd International Conference on High Performance Computing & Communications; 7th International*

Conference on Data Science & Systems; 19th International Conference on Smart City; 7th International Conference on Dependability in Sensor, Cloud & Big Data Systems & Application.