Improving Enterprise Scale Test Automation with ML-Based Predictive Analytics

Gumeniuk D., King, T. M.

Dmitriy_Gumeniuk@epam.com

Abstract

AI and machine learning continue to make considerable progress in advancing state of the art computing. Recently, the focus has been on generative AI systems like ChatGPT. However, these models encompass a fundamental aspect of ML, the ability to observe and analyze historical data and make predictions. Automation is a core component of any testing strategy but is especially important in enterprise projects with hundreds of thousands of tests. Collecting, coalescing, and analyzing the results of large-scale test automation projects across multiple levels and types of testing is challenging. However, incorporating AI and ML into the testing process, including leveraging it to enhance test reporting and analytics, is an effective way to improve large scale test automation. This paper describes some of the key challenges faced in large scale test automation projects and provides a summary of how AI and ML are being applied in practice to overcome those challenges. Emphasis is placed on the stability of executing automated tests at scale, and providing meaningful, actionable insights from the results. The findings and experiences from enterprise case studies are summarized, including one that leverages an open-source, AI-powered test automation dashboard.

Biography

Dmitriy Gumeniuk is the Head of Testing Products at EPAM Systems, a leading global provider of digital platform engineering and software development services. With over 17 years of experience in software development, Dmitriy has been at the forefront of innovation in test automation. He has led the development of multiply solution and accelerators at EPAM's Test Competency Center, focusing on the application of Machine Learning and Neural Networks in test automation. Dmitriy is a frequent presenter at conferences, has notable contributions into open-source community and DevTestOps community by organizing the annual DelEx conference.

Tariq King is the Vice President of Product-Service Systems at EPAM, where he manages a portfolio that lies at the intersection of software products and services, and supports the business through technology consulting. He has over 15 years' experience in software engineering and testing and has formerly held positions as Chief Scientist, Head of Quality, Director of Quality Engineering, Manager of Software Engineering and Test Architect. Tariq holds Ph.D. and M.S. degrees in Computer Science from Florida International University, and a B.S. in Computer Science from Florida Tech. He has published over 40 research articles in peer-reviewed IEEE and ACM journals, conferences, and workshops, and has been an international keynote speaker at leading software conferences in industry and academia.

1 Introduction

In the rapidly evolving landscape of software development, test automation has emerged as a cornerstone for ensuring software quality, particularly in large-scale enterprise projects. While automation has significantly improved the efficiency and coverage of testing, it also brings forth a set of challenges, especially when hundreds of thousands of tests are involved. The complexity of managing and analyzing test results at this scale necessitates innovative solutions.

Artificial Intelligence (AI) and Machine Learning (ML) have shown promise in various domains, including natural language processing, computer vision, and more recently, in software testing. This paper focuses on the application of ML-based predictive analytics to improve enterprise-scale test automation. Specifically, we explore the use of k-Nearest Neighbors (kNN) models, enhanced by Boosting Trees algorithms, for the triage of failed test reports. The primary objective of this work is to automate the categorization of failed test reports into distinct groups: Product Bugs, Automation Issues, and System Issues. Such categorization aids in streamlining the workflow for engineers and provides actionable insights for decision-makers.

This paper outlines the key challenges associated with large-scale test automation and offers an AI-driven solution to overcome some of these challenges. The proposed methodology incorporates text normalization techniques to reduce noise in logs and gives a higher weight to recent test results to produce a more relevant triage. The insights are grounded in real-world case studies, highlighting the tangible benefits of integrating AI and ML into test management systems. The paper is organized as follows: Section 2 provides the necessary background and outlining the problem statement, and Section 3 summarizes the key challenges that motivate this work. Section 4 describes our approach and the methodology employed, followed by a detailed explanation of the technical aspects of the ML models leveraged in the solution. Section 5 presents real-world case studies, including an analysis applied improvements and application to validate the effectiveness of our approach. In the last section, the paper concludes with a discussion, future work, and acknowledgments.

By the end of this paper, the reader will understand the challenges in large-scale test automation and how ML-based predictive analytics can offer practical solutions to overcome them.

2 Background

Continuous Testing and Quality Gates

Test automation has been an integral part of software development for years, enabling teams to execute a large number of tests efficiently. However, as software systems grow in complexity and scale, traditional test automation methods often fall short. This is especially true as full transparency into the reasons for test failures has become integral to establishing Continuous Testing (CT) practices with automated decisions on Quality Gates (QG). The ability to fully triage all failed test reports is crucial for making GO or No-Go decisions at Quality Gates — points in the delivery pipeline where the product's quality is assessed. Fully triaged failed reports provide clarity on the number of existing product issues versus test automation issues, thereby influencing critical decisions in the CT process.

AI and ML in Test Automation

Artificial Intelligence (AI) and Machine Learning (ML) have made significant strides in various domains, including software testing. While much of the focus has been on generative models and natural language processing, predictive analytics has received less attention. Predictive analytics in software testing involves using historical data to make future predictions, such as the likelihood of a test failing or the probable cause of a failure. Enterprise-level projects often involve hundreds of thousands of tests, making the management and analysis of test results a daunting task. The challenges include, but are not limited

to, the triage of failed tests, identifying flaky tests, and providing actionable insights for decision-making. The triage of failed test reports is a critical aspect of test automation. Traditionally, this has been a manual process, requiring significant effort from engineers. The application of k-Nearest Neighbors (kNN) models, enhanced by Boosting Trees algorithms, offers a promising automated solution. These algorithms use logs and failed stack-traces to form predictive models that can categorize failed tests into groups like Product bugs, Automation issues, and System issues. The initial categorization of a failed test report often falls under a "To Investigate" state, requiring manual intervention. ML algorithms can automate this initial triage, thereby streamlining the workflow for engineers and reducing the manual effort involved.

3 Challenges in Enterprise Scale Test Automation

Automated testing is now a necessity in enterprise software development, especially for agile or DevOps environments that prioritize speed and continuous delivery. Yet, while automation can streamline many aspects of testing, it also brings forth a unique set of challenges that are amplified at an enterprise scale. In this section, we explore some of these obstacles that organizations often encounter in implementing and sustaining automated testing at a large scale.

Complexity of Test Scenarios. One of the primary challenges lies in the complexity of enterprise-scale software. Large-scale applications often have myriad functionalities and subsystems that need rigorous testing. Creating automated test scenarios that accurately mimic real-world use cases becomes exponentially more difficult as the application grows in complexity.

Management of Test Data. Data is the backbone of any testing regime. In an enterprise setting, handling vast amounts of test data across different databases, services, and environments becomes a laborious task. Ensuring that the data is consistent, secure, and up-to-date for every test cycle is often easier said than done.

Integration with Existing Systems. Enterprises usually have a blend of modern and legacy systems. The automated testing tools need to be compatible with different architectures, APIs, and even other testing tools. The integration often demands extensive customization and fine-tuning, making it a time-consuming endeavor.

Resource Constraints. Resource allocation is often constrained by budgetary limits, existing project commitments, and availability of skilled personnel. Building and maintaining an automated testing suite can be resource-intensive, often requiring investment in specialized expertise and ongoing training.

Maintainability of Test Scripts. Over time, as the software evolves, the automated test scripts can become outdated or irrelevant. Updating these scripts to adapt to new features, changes in UI, or alterations in workflow can be an overwhelming task, especially for large codebases.

Noise and False Positives. Automated tests are not immune to generating noise in the form of false positives or inconclusive results. Filtering out these anomalies to focus on genuine issues requires additional layers of analysis, which can sometimes offset the time saved through automation.

Evolving Technologies. The technology landscape is ever-changing. The introduction of new development methodologies, languages, and tools can make existing automation frameworks obsolete. Organizations need to be agile enough to adapt to these changes without disrupting their existing testing pipelines.

Understanding the ‘Why’ Behind Failures. Traditional automated systems are excellent at pointing out 'what' has failed but are usually not designed to diagnose 'why' it has failed. This requires human intervention, further complicating the automation process.

By understanding these challenges in depth, we can better appreciate the value that intelligent, adaptable auto-analysis brings to the table. In the next section, we will examine how this tool addresses some of these challenges by leveraging machine learning and advanced analytics.

4 Machine Learning in Test Automation: A Paradigm Shift

The challenges presented by enterprise-scale test automation are not just hurdles; they are opportunities for innovation. One promising avenue is the application of machine learning algorithms to the automation process, a shift that significantly redefines our approach to testing. This section aims to elucidate the role that machine learning plays in modern test automation.

Intelligent Test Data Management. Traditional test data management systems struggle with the complexity and volume of enterprise data. Machine learning algorithms can predict what kinds of data are most likely to induce failure, allowing for more targeted testing scenarios. ReportPortal, for instance, uses analytics to understand the past behavior of test data, helping in creating more realistic test cases.

Self-Healing and Adaptive Test Scripts. One of the most significant challenges in maintaining automated test suites is the requirement for continual updates. Self-healing mechanisms can dynamically identify the correct locator at the moment of test execution. When a script encounters a missing or changed element, the self-healing algorithm kicks in to search for the most likely alternative locator based on historical data or predefined heuristics. This not only minimizes test flakiness but also reduces the manual effort required to update test scripts, further enhancing the adaptability and resilience of automated test suites. For the adaptive test scripts Machine Learning models can learn from test outcomes to automatically adjust test scripts, all this dramatically reducing the overhead involved in maintenance.

Enhanced Failure Diagnosis. While traditional test automation tools can identify a failure, they usually can't explain why it failed. Advanced machine learning models can analyze a multitude of variables in real-time to offer not just a 'what' but a 'why,' thereby significantly aiding in troubleshooting.

False Positive Identification. Machine learning algorithms can provide an additional layer of analysis to distinguish between actual defects and false positives, a common issue in automated testing. This significantly reduces the time required for manual oversight.

Customizable and Scalable. The use of machine learning in testing allows for more personalized, project-specific testing scenarios. It adapts to the idiosyncrasies of a given software ecosystem, thereby offering a higher level of customization without sacrificing scalability.

Real-World Applications: Test Failure Categorization. ReportPortal has integrated machine learning to automatically categorize defects and suggest similar historical defects. Its auto-analyzer feature includes retraining capabilities that take into account human-made decisions over time. The tool leverages machine learning to analyze failure reports and identify current patterns in stack traces, thereby making the tool more intelligent and adaptable over time.

By incorporating machine learning into test automation, enterprises can not only overcome many of the challenges previously discussed but also open up new avenues for optimization and quality assurance.

5 Case Study

ReportPortal is an open-source solution that offers a practical method of how ML can be applied to test automation. The platform uses advanced algorithms for auto-analysis and provides options for manual analysis, thereby offering a balanced approach.

5.1 Experimental Set Up

To gauge the efficacy of machine learning algorithms in a structured methodology was applied. The study adopted a longitudinal analysis approach, evaluating 20 large-scale enterprise projects over a span of 6 months. The projects selected for this study were diverse in nature, ranging from API testing to end-to-end UI testing, including BDD, and showcased a diverse mix of programming languages such as Java, Python, JavaScript, and C#. Additionally, different test runners, including TestNG, Junit, NUnit, Pytest, Jasmine, and MochaJS were utilized. Another layer of complexity was added by the various speaking languages used in logs to elucidate causes of failures (English, Russian, Chinese).

Data Sources

For study, we used and obfuscated production database with total data amassed as 2.7Tb.

The primary types of data were collected:

1. **Automated Testing Results**: Test execution logs were collected, including logs of test automation execution, specifically all error logs and stack traces of the failed test cases, test case IDs, parent test suites, test parameters and attributes.

2. **Manual Reviews**: To serve as a benchmark, manual categorizations were also conducted for a subset of tests, focusing on the areas where machine learning-based recommendations were provided.

Metrics for Evaluation

The performance of machine learning in automated testing was evaluated based on the following metrics:

- **Accuracy**: The ratio of correctly classified defects to the total number of defects.
- **Precision**: How many of the flagged issues were indeed defects.
- **Recall**: Of all the defects, how many were correctly identified by the machine learning algorithms.
- **F1-Score**: A balanced metric that considers both precision and recall.
- **Time-Saved**: The amount of manual effort saved by automating defect categorization and recommendation processes.

Tools and Technology

ReportPortal: This was the primary tool used for this study, since it serves a role of a single entry point for the test reporting in our organization and consolidate a variety of testing results.

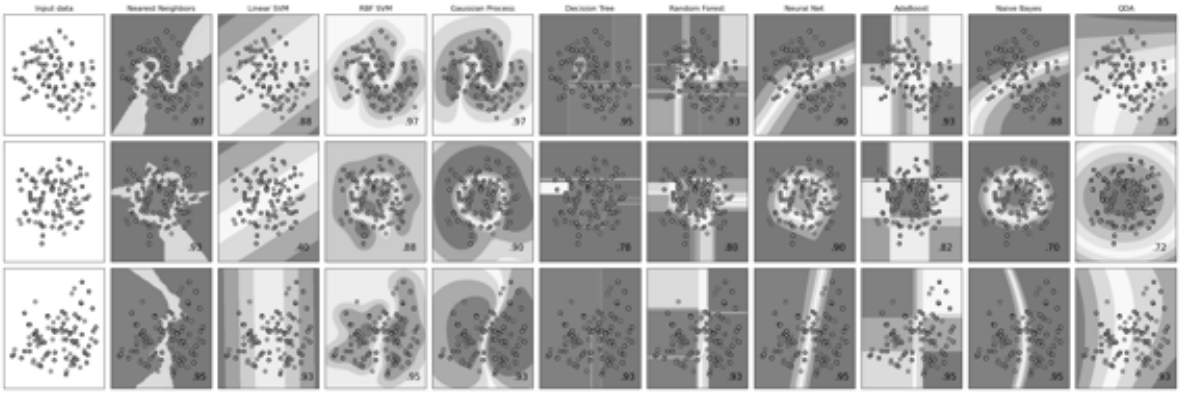
Python for Data Analysis: Python libraries such as Pandas and Scikit-learn were used for data pre-processing and statistical analysis.

Tableau for Visualization: Key findings were represented graphically through Tableau dashboards.

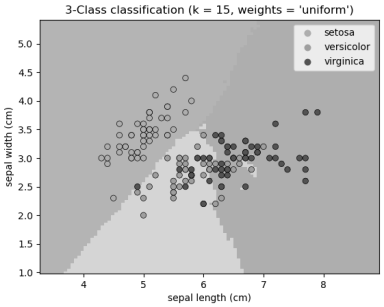
Statistical Methods

To ensure the robustness of our findings, statistical tests like chi-square tests for independence, t-tests for comparing means, and logistic regression for understanding variable impact were employed.

By adhering to this comprehensive methodology, we aimed to ensure the reliability and validity of our findings. The results are presented in the subsequent section.



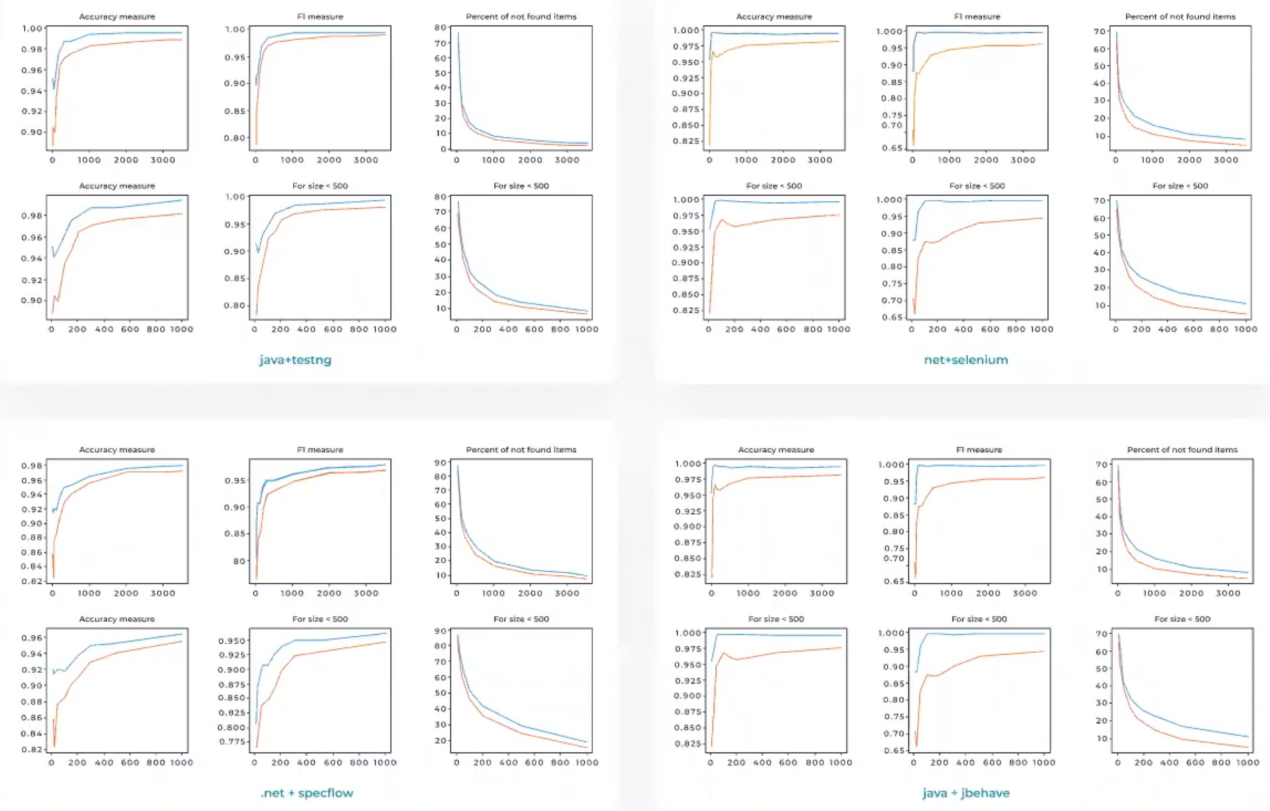
classifications



k-Nearest Neighbors algorithm representation

5.2 Results and Analysis

Through an exhaustive examination of 20 large-scale enterprise projects, this study sought to demystify the complexities and challenges of test automation at an enterprise level. The data amassed was comprehensive, offering a rich dataset for in-depth analysis. Here are the pivotal findings:



Log Quality and the Human Factor

- The quality of logs plays a crucial role in the outcomes of machine learning algorithms. Generic error logs or assertion error messages, such as "*expected <true>, but was <false>*" or "*failed*", hinder algorithms from deducing meaningful insights.

- A mere enrichment of error descriptions in assertions or exceptions managed by try-catch led to an 80% surge in categorization accuracy.

- The principle is clear: If a QA engineer unfamiliar with the failing test can't discern the issue based on the log description, machine learning will likely struggle as well.

The Human Factor: "Friday Bug". Human uncertainty in categorizing test failures often leads to the selection of inappropriate categories. These categories often undergo transformation after 2-3 test runs. The "Friday Bug" anomaly was identified, where engineers, in a bid to wrap up their week, hastily marked a segment of failed tests as defect-free. This behavior significantly blurred clustering and reduced precision.

Incremental Learning is essential due to changes in the project landscape, such as the introduction of new tests, application evolution, and enhanced coverage.The machine learning model should reuse newly marked data with the latest categorizations made both by human and machine input.

Decisions Based on Recency. For the ML model it does make sense to prioritize latest categorization decision over the old ones via boosting the scoring. Decisions related to tests were heavily influenced by the recency of the decision timestamp. As teams collectively unearth details, decisions about functionalities evolve.

Engineer Decisions versus Machine Decisions should always have a priority. However, the over-reliance on human decisions can sometimes adversely impact accuracy, especially when considering the human element discussed in the "Friday Bug" anomaly.

Traditional automated systems are excellent at pointing out 'what' has failed but are usually not designed to diagnose 'why' it has failed, necessitating human intervention.

5.3 Stability and Actionable Insights

In the realm of test automation, especially at the enterprise scale, stability and the ability to derive actionable insights are paramount. The integration of machine learning into this domain has brought forth a new dimension of stability and has enhanced the depth and breadth of insights that can be extracted. Here's a detailed exploration:

5.3.1 Stability in AI/ML-Driven Solutions:

- **Consistency in Predictions**: Machine learning models, once trained with a substantial amount of data, tend to provide consistent predictions. This consistency is vital for test automation, ensuring that the same input will yield the same categorization or recommendation, barring any model retraining.

- **Adaptability to Changes**: One of the hallmarks of machine learning models is their ability to adapt. With incremental learning, as discussed in the previous section, models can adjust to the evolving nature of applications and their testing scenarios. This adaptability ensures sustained stability even in dynamic environments.

- **Reduced Human Error**: By automating certain decision-making processes, the potential for human errors, such as the "Friday Bug" anomaly, is significantly reduced. This leads to a more stable and reliable testing process.

5.3.2 Deriving Actionable Insights:

Root Cause Analysis: Advanced machine learning models can delve deeper into test failures, not just identifying the 'what' but also illuminating the 'why'. This deeper understanding aids in troubleshooting and can guide corrective actions more effectively.

Predictive Analysis: Beyond just analyzing current test results, machine learning models can predict future outcomes based on historical data. For instance, predicting which tests are more likely to fail in the next cycle or identifying potential bottlenecks in the testing pipeline.

Optimization Recommendations: By analyzing patterns over time, machine learning can suggest optimizations in the testing process. This could range from recommending changes in test sequences to optimize execution time or suggesting areas where test coverage could be enhanced.

Feedback Loop for Continuous Improvement: The integration of machine learning creates a feedback loop. As tests are executed and results are analyzed, the insights derived can be fed back into the system, leading to continuous refinement and improvement of both the testing process and the machine learning models.

5.3.3 Real-World Application and Challenges

The ReportPortal solution exemplifies the practical application of deriving actionable insights. Its auto-analyzer feature not only categorizes defects but also suggests similar historical defects. This dual functionality provides immediate insights into potential defect patterns and historical context, aiding in faster resolution.

While the integration of machine learning offers enhanced stability and insights, it's essential to recognize that it's not a panacea. The quality of insights is heavily dependent on the quality of data fed into the system. As highlighted in the "Results and Analysis" section, generic or non-descriptive logs can hinder the depth of insights derived.

Additionally, while machine learning models offer stability, they need periodic retraining to maintain their accuracy and relevance, especially in rapidly evolving application landscapes.

In conclusion, the fusion of machine learning with test automation at the enterprise scale offers a promising avenue for enhanced stability and deeper, more actionable insights. However, its success hinges on the quality of data, the relevance of the models, and the continuous feedback loop that ensures both the testing process, and the models evolve in tandem.

6 Conclusion and Future Direction

The integration of machine learning into enterprise-scale test automation has already shown significant promise, but the journey is just beginning. As technology continues to evolve and the demands of software testing grow more complex, there are several potential directions that this fusion of AI/ML and test automation might take. Here's a glimpse into the possible future trajectories:

6.1 Concluding Remarks

The landscape of software testing, particularly at the enterprise scale, is undergoing a transformative shift. As the demands of software quality assurance grow in complexity and scale, traditional methods of test automation are increasingly proving inadequate. This research has delved deep into the challenges faced by large-scale test automation and has illuminated the potential of machine learning as a solution to many of these challenges.

Several key takeaways emerge:

1. **Human Factor in Automation**: While automation aims to reduce human intervention, the quality of human input, especially in the form of detailed logs and accurate categorization, remains paramount. The "garbage in - garbage out" principle holds true, emphasizing the need for quality data for effective machine learning outcomes.
2. **Machine Learning's Potential**: The integration of machine learning into test automation offers a plethora of benefits, from intelligent test data management and self-healing test scripts to generation of the test case scripts. Tools like ReportPortal showcase the tangible benefits of this integration, offering both auto-analysis and manual analysis capabilities.
3. **Expanding Data Sources**: The potential data sources for analysis and categorization are vast. Application logs, often overlooked, can provide deeper insights into system behavior. Additionally, the advent of Visual AI, leveraging image recognition techniques, can further enhance the depth and breadth of analysis.
4. **Continuous Evolution**: The technology landscape is ever-evolving. As new challenges emerge, the fusion of AI/ML and test automation will need to adapt. Continuous research, learning, and adaptation will be pivotal in ensuring that this integration remains relevant and effective.

In conclusion, the fusion of machine learning and test automation represents a promising frontier in the realm of software quality assurance. While challenges abound, the potential benefits in terms of efficiency, accuracy, and quality make this a journey worth undertaking. As technology continues to evolve, it is incumbent upon the software testing community to harness the power of machine learning, continually innovate, and set new benchmarks in software quality.

6.2 Future Directions

Advanced Machine Learning Models:

- **Deep Learning Integration**: While the current focus has been on predictive analytics using models like k-Nearest Neighbors (kNN) and Boosting Trees algorithms, there's potential to integrate deep learning models, especially for more complex pattern recognition in test logs and application under test logs.

- **Natural Language Processing**: Advanced NLP techniques can be employed to better understand and categorize error logs, especially those written in natural language or in multiple languages, as highlighted in the methodology section.

Application Logs as a Rich Data Source, often overlooked, can serve as a treasure trove of information. These logs can provide deeper insights into system behavior, user interactions, and potential error patterns. Machine learning models can be trained to analyze these logs, offering a more granular understanding of system performance and potential areas of concern.

Visual AI for Enhanced Analysis, as an Image recognition technique can be employed to detect error patterns or pop-ups in application screenshots. This "Visual AI" can be pivotal in identifying visual discrepancies that might be missed in traditional testing. Furthermore, pixel-perfect change detection can ensure that any visual modifications to the application, intentional or otherwise, are immediately flagged and reviewed.

Automated Test Generation with Generative Large Language Models, aimed at a language generation**,** not just analyze but also generate test cases based on software requirements, user behavior, and historical defect data. This can lead to more comprehensive and targeted test coverage.

Self-Optimizing Test Suites beyond self-healing test scripts, envision test suites that can self-optimize, reordering tests based on dependencies, historical fail rates, or execution time, ensuring the most efficient test run possible.

Enhanced Integration with Development and Operations, allowing for real-time feedback loops. This can lead to immediate defect resolution during the development phase itself, further streamlining the DevOps pipeline.

Augmented Reality (AR) and Virtual Reality (VR) Testing become more mainstream, there will be a growing need for automated testing in these domains. Machine learning can play a pivotal role in understanding and simulating human interactions in virtual environments.

1. **Ethical Considerations and Bias Mitigation**

- As machine learning models make more decisions, there will be a growing need to ensure that these decisions are free from biases, especially if they impact user experiences or business outcomes. Future

research might delve deeper into ensuring fairness, transparency, and accountability in AI-driven test automation.

2. **Continuous Learning and Adaptation**

- The models will need to be designed for continuous learning, adapting not just based on historical data but also real-time data streams, ensuring that the testing process remains agile and responsive to immediate software changes.

3. **Collaborative AI in Testing**

- Envision a future where AI-driven testing tools collaborate, sharing insights, models, and best practices across different platforms and organizations, leading to a collective improvement in software quality globally.

In essence, the future of machine learning in enterprise-scale test automation is brimming with possibilities. While the challenges are manifold, the potential benefits in terms of efficiency, coverage, and quality assurance make this a frontier worth exploring in depth. As with any technological evolution, continuous research, experimentation, and adaptation will be key to realizing the full potential of this integration.

7 Acknowledgments

We extend our deepest gratitude to the team members of ReportPortal for their invaluable contributions and relentless dedication to the project. Their expertise and commitment have been instrumental in the success of our research and the development of the platform.

Special thanks go to key figures in the machine learning community, whose pioneering work laid the foundation for our research. Their insights and innovations have been a constant source of inspiration.

We are particularly grateful to Maryia Ivanina and Andrei Varabyeu for their guidance, mentorship, and unwavering support throughout this project. Their expertise has been invaluable in navigating the complexities of machine learning in test automation.

Lastly, we would like to express our appreciation to the members of EPAM's software testing practice. Their contributions, both big and small, have enriched this work in countless ways. Their collective wisdom and experience have been indispensable in shaping the research and ensuring its relevance to the industry.

8 References

Briggs, K.A., King, T.M. "Semi-autonomous, site-wide A11Y testing using an intelligent agent." Journal of Software Testing, Vol. 12, No. 3, 2020.

Daye, Philip Jr. "Quality Guild: Creating a Culture." Journal of Software Quality, Vol. 10, No. 2, pp. 25-40, 2019.

Cover, T. and Hart, P. "Nearest neighbor pattern classification." IEEE Transactions on Information Theory, Vol. 13, 1967.

Shubert, E., Rousseeuw, P.J., “Faster k-medoids clustering: improving the PAM, CLARA, and CLARANS algorithms.” https://arxiv.org/abs/1810.05691 (accessed August 27, 2019).

Pawlik, M., Augsten, N. 2015. “Efficient Computation of Edit Tree Distance.” ACM Transactions on Database Systems, vol. 40 (1), No. 3. https://dl.acm.org/citation.cfm?id=2699485, (accessed August 27, 2019).

Freund, Y. and Schapire, R.E. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting." Journal of Computer and System Sciences, Vol. 55, 1997.

"ReportPortal: An Open-Source Reporting Platform with Machine Learning Capabilities." Official Documentation, 2022. https://reportportal.io/docs/

Pedregosa, F. et al. "Scikit-learn: Machine Learning in Python." Journal of Machine Learning Research, Vol. 12, 2011.

"Tableau Software." Official Documentation, 2022. https://help.tableau.com/current/pro/desktop/en-us/gettingstarted_overview.htm

Witten, I.H., Frank, E., Hall, M.A., and Pal, C.J. "Data Mining: Practical Machine Learning Tools and Techniques." Morgan Kaufmann, 2016.

"EPAM Systems: A Case Study in Large-Scale Test Automation." Internal Whitepaper, EPAM Systems, 2022.

McKinney, Wes. "Python for Data Analysis." O'Reilly Media, 2018.

"Challenges and Solutions in Large-Scale Test Automation: A Survey." Journal of Software Engineering Research and Development, Vol. 8, No. 1, pp. 1-20, 2020.

"Continuous Testing and Quality Gates: A Practical Guide." Journal of Software Testing, Vol. 14, 2019.