https://doi.org/10.53656/igc-2025.18

IMPLEMENTATION OF MACHINE LEARNING MODELS TO PREDICT FAILURE RATE OF RPA PROCESSES

Yanka Aleksandrova
University of Economics, Varna (Bulgaria)
Mihail Radev
University of Economics, Varna (Bulgaria)

Abstract. This paper presents a practical framework to extract and engineer process-execution elements of Power Automate logs and to train ML models to predict flow failures. We compare Logistic Regression, Random Forest, XGBoost, Voting, and Stacked ensembles; the Stacked ensemble is the best (ROC-AUC = 0.941). The most influential factors, which are highlighted by feature importances and SHAP, include complexity, load, retries, SLA deviation and reveal bottlenecks overall and across processes.

Keywords: robot process automation, machine learning, process monitoring framework

JEL: C45, C53, M15

Introduction

In modern business automation, robotic process automation (RPA) has emerged as a technology that allows organizations to automate repetitive, rule-based tasks. With the growth of RPA software robots and automated workflows in organizations, it is becoming increasingly important to effectively monitor these automated workflows to ensure high productivity, minimize downtime, and quickly resolve potential issues. The use of machine learning models can enable the prediction of failures and potential problems in automated processes.

Robotic Process Automation (RPA) improves efficiency and reduces the cost of operations for organizations. Accurately predicting the failure rate of RPA systems is crucial for maintaining the stability of workflow operations, controlling costs associated with them, improving their performance, and creating more stable and reliable subsequent workflows. Since run flow status prediction is not an isolated task, a comprehensive framework for predicting the failure rate of RPA executions is needed to reduce the operational risk of delays and data inconsistencies caused by

failures in the software robots running in the organization. By predicting the failure rate of flows, organizations receive an early warning mechanism for proactive and timely resolution of errors. Early identification prevents cascading failures in other processes, ensuring business continuity and service level agreements (SLAs). The overall goal of implementing such predictive framework is to achieve improvement in process design and reduce and optimize routine work in organizations. Additional results from the implementation are related to a substantial cost minimization due to a decrease in potential downtime and frequency of failures. Important results that should be expected include also improvement in the scalability of processes and their complexity, facilitating their management and documentation.

Literature Review

Failure prediction is concerned with predicting whether a material system will fail at a certain point in the future. This task is part of industrial support strategies, such as predictive support. Machine learning (ML) technology is increasingly being used to solve this task. A detailed review of research results in the field of using machine learning (ML) technology for failure prediction in industrial support is given in the publication of Leukel et al. (2021, p. 87). The practical and theoretical foundations of such predictive systems are further explored in recent applied studies on digital transformation (Biolcheva & Shterev, 2024).

Failure prediction approaches for industrial maintenance are like the prediction of problems and failures in robotic automated processes (RPA). Current and historical data representing system states, events, and operations are analyzed. The use of machine learning (ML) allows training a predictive model from time series data, evaluating the model's performance, and deploying it in a production environment (Hall & Rasheed, 2025). This methodology is in line with new research emphasizing digital decision-making (Antonova et al., 2024).

There is evidence showing that machine learning (ML)-based failure prediction models are effective for various systems. A study by Iosif et al. (2025) shows the use of prediction models in agriculture – in biotechnical systems. There are studies in the field of industrial equipment and machinery (Majid & Anuar, 2022; Karuppusamy, 2021) and a publication by Patrício et al. (2025), proving the hypothesis that the combined application of RPA and ML significantly improves the ability to predict failures. The models can also be used in the automotive industry (Canciglierie et al., 2021; Wewerka & Reichert, 2020). They also have applications in ICT – in cloud services and computing (Mallreddy & Vasa, 2024). The increasing interdisciplinarity of predictive analytics also brings with it an increasing importance of aligning machine learning research with digital innovation frameworks (Minkov & Ivanov, 2024).

Various Machine Learning algorithms have been used in published studies. ML models such as Support Vector Machines (SVM), Random Forest, Decision Tree

and ensemble methods achieve high prediction accuracy when applied to failure prediction tasks in various automated and cloud environments.

The Random Forest model has been identified as the best performing model for predicting failures in complex manufacturing environments according to a publication by (Ojeda et al., 2025). The authors consider a predictive model based on machine learning in relation to the automotive industry and verify its ability to reduce the impact of unplanned downtime. As a result of the study, the Random Forest model has been identified as a highly effective prediction tool, combining strong predictive performance with practical benefits in reducing downtime.

In a similar direction for the use of this method are the results of the study on the use of predictive models for predicting cloud failure. The publication is by Tengku Asmawi (Tengku Asmawi et al., 2022) For predicting task failure, the Decision Tree and Random Forest models were derived as the most suitable. Datasets with different characteristics were used: real datasets, real and synthetic datasets.

Methodology

The methodology used in this research is aimed at applying machine learning models to predict process flow status and identify potential bottlenecks and problems hindering the successful execution of processes. Data is extracted from Power Automate logs using end-to-end data framework. The log files, in which events are registered during the execution of the flow, contain time series and event data that comprehensively reflect the actions of the software robot, its execution duration, and the characteristics of the environment. The steps of data extraction, enrichment and aggregation are shown in Figure 1.

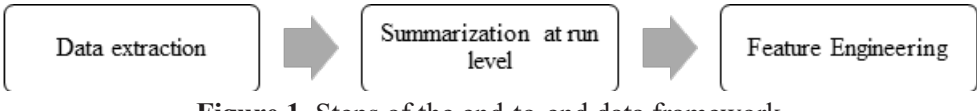


Figure 1. Steps of the end-to-end data framework

On the first step "Data extraction" Power Shell scripts are developed and implemented to continuously extract data from Power Automate log files. The generated dataset contains 2000 observations for the execution of 5 different processes over a period of 20 days.

Data include unique identifier for every flow run (RunID), the name of the process flow (FlowName), a timestamp when the flow execution started (StartTime), number of steps in the process (StepCount), execution time in seconds (TortalDurationSec), duration in seconds of the longest-running step (MaxStepDuration), number of retries performed during the execution of the flow run (Retries), binary flags indicating whether the process flow has a parallel branch (HasParallelBranch) or a condition (HasCondition). The target variable (Failure) is also extracted from log

files. It has two possible values – 0 (the flow run is executed successfully) and 1 (the flow run exited with an error).

The goal of the second step from the end-to-end framework, "Feature Engineering" is to add more valuable information to the extracted data in order to empower machine learning models to discover hidden patterns and relationships between factor and target variables. Engineered features can be classified into the following categories.

- Features related to the flow run execution:

– AvgStepDuration – average execution time per step

– RetryRate – number of retries divided by step count. As process flow with more steps are more likely to be retried during execution, this feature will calculate number of retries per step, decreasing the influence of the step count.

– DurationPerStep – average duration of one step from the flow run

- Features related to the bottleneck step:

– BottleneckRatio – share of the total runtime consumed by the bottleneck step.

– MaxVsAvgStepRation – ratio of the bottleneck step duration compared to the average step duration in seconds. This added feature acts as an outlier for the slowest step in the flow run.

- Features related to the history of the flow run executions:

– Hour – hour extracted from StartTime

– DayOfWeek – day of week

– IsWeekend – binary flag indicating whether the run occurred on weekend (1) or no (0)

– IsPeak – binary flag indicating whether the run occurred during a business-defined peak window (1) or no (0)

– FlowFreq – the time pattern of the process flow indicating number of times this flow runs weekly

– ExpectedDuration – estimated normal duration for the flow based on past historic data

– SLADeviation – the difference between total duration of the current flow run and estimated normal duration

– SLAOverrunFlag – a binary flag showing if SLADeviation exceeded the set tolerance (1) or no (0)

– FlowRollingFailRate3 - Rolling average of the failure for the last 3 runs of the same flow. It is assumed that process flows with past failures are more likely to fail in future.

Results

After the data is extracted, enriched and summarized, machine learning models are trained to predict process status. Models are built using best performing classification algorithms like random forest, XGBoost, Voting and Stacked

ensembles. The data preparation before models training includes transforming the categorical features into numerical values using OneHotEncoder and StandardScaler and SimpleImputer to numerical values. To address the class imbalance, various appropriate methods for random oversampling have been implemented, since the class distribution is highly imbalanced with approximately 25% failed and 75% successful flow run executions. Across different processes the fail rate varies from 17.7% to 34.5%. Hyperparameter optimization is also applied within the set parameter sets as optimal parameter values can significantly improve the models' performance.

All trained models are scored against one and the same test dataset to ensure comparability. Evaluation results are presented in Table 1.

Table 1. Model evaluation results

Model	ROC_AUC	PR_AUC	Sensitivity	Specificity	Accuracy	F1 score
Random Forest	0,9365	0,8662	0,8255	0,9180	0,8950	0,7961
XGBoost	0,9347	0,8644	0,8188	0,9135	0,8900	0,7871
Stack Ensemble	0,9408	0,8726	0,8456	0,8958	0,8833	0,7826
Vote Ensemble	0,9398	0,8710	0,8255	0,9157	0,8933	0,7935

From the comparative analysis of the performance of the trained model (see Table 1), several conclusions can be drawn. First, all four models achieve consistently high predictive power, with Receiver Operation Characteristics Area Under Curve (ROC-AUC) values above 0.93 and Precision-Recall Area Under Curve (PR-AUC) values above 0.86. These values indicate reliable detection of failures and prove the efficiency of the data balancing techniques applied. Second, the Stacked ensemble demonstrates the highest sensitivity (0.8456) and highest PR-AUC (0.8726), making it the most effective option when the key objective is to detect run flows which are more likely to fail, even at the cost of slightly lower specificity. Third, Random Forest shows the highest specificity (0.9180) and the best F1 score (0.7961), establishing it as the most balanced classifier, suitable for minimizing false positives while maintaining sufficient sensitivity. The Voting ensemble consistently ranks just below Random Forest across most metrics, offering robustness and stability as a general-purpose model. XGBoost, while demonstrating stable results, does not predominate in any single metric and thus appears less competitive as a standalone classifier.

Based on these findings, the choice of which model to be deployed should be aligned with organizational priorities. If the primary goal is operational safety and

risk minimization then the preferred model should be Stacked ensemble because of its superior sensitivity and PR-AUC performance. At the same time, a system relying on this model will accept more false alarms in order to avoid missed potential failure. On the other hand, if the focus is on efficiency and reducing false positives, Random Forest is the optimal model, as it balances sensitivity with relatively high specificity and F1 score. If both objectives are equally important and the primary organization goal is robustness across varying conditions, the Voting ensemble provides a good alternative as a combination of strengths of multiple classifiers. A viable approach is also to implement a flexible strategy depending on the assessed environment risk and estimation of the costs associated with false positives and false negatives. Such strategy will consider using Stacked ensemble in risky environments where unidentified failures carry severe costs, and Random Forest or Voting ensemble in lower-risk environments.

Exploring the feature importances for tree-based machine learning models can be used to identify the most influential factors contributing to the status of the run flows. A side-by-side comparison of the top 10 feature importances is shown on figure 2.

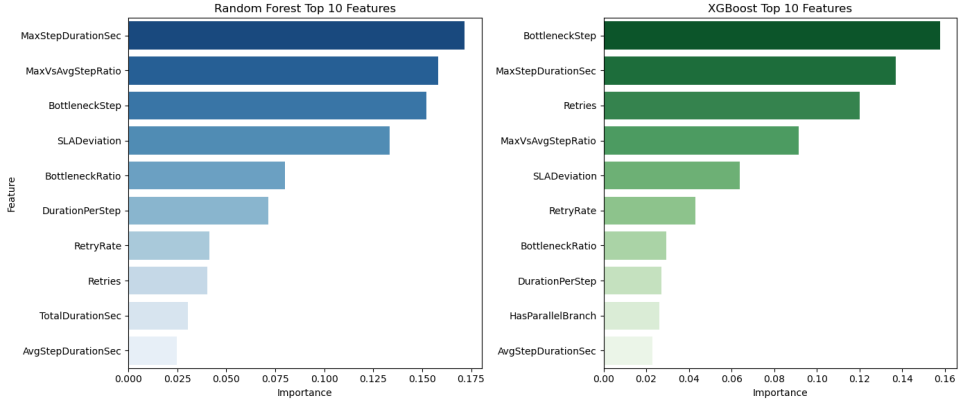


Figure 2. Comparison of top 10 feature importances between Random Forest and XGBoost models

The two models – Random Forest and XGBoost – demonstrate overlapping but slightly different views on how important the different factor variables on predicted run flow failure are. In the Random Forest model, the most influential features are MaxStepDurationSec, MaxVsAvgStepRatio, BottleneckStep, and SLADeviation. Together, these four features account for more than half of the total importance. This suggests that the model heavily relies on measures related to process timing, particularly the maximum duration of individual steps, how this compares to the

average, and whether a bottleneck step exists. SLA deviation further strengthens the interpretation that timing-related inefficiencies are critical warning signals for failure. Another relevant factors include also BottleneckRatio, DurationPerStep, and error-related metrics such as RetryRate and Retries.

The other model, XGBoost, ranks the features slightly differently, giving the highest importance to BottleneckStep, followed by MaxStepDurationSec and Retries. This indicates a stronger emphasis on bottleneck step presence and explicit measuring of retry counts, which directly capture problematic execution patterns. While Random Forest prioritizes duration ratios, XGBoost appears more sensitive to indicators of system instability and repeated execution attempts. It's worth mentioning that the feature "HasParallelBranch" appears in XGBoost's top ten but is absent from Random Forest's most influential factors, suggesting that XGBoost gives preference on structural complexity of the process, where parallel execution increases risk of failure.

From these results, a few conclusions can be drawn. Both models agree that bottlenecks and timing deviations are central to predicting failures, confirming the initial observations that delays and imbalances in execution flow are possible causes for problems. Random Forest tends to emphasize on comparative duration features, while XGBoost highlights operational stress measures such as retries and branching. This suggests that the two models can be viewed as complementary: Random Forest is effective at capturing gradual process inefficiencies, while XGBoost is better at identifying important execution problems. The appearance of parallel branching in XGBoost's top features indicates that complexity-related indicators should not be ignored in feature engineering, as they may gain importance in other contexts or under different data distributions.

From a modelling perspective, combining both sets of feature insights in an ensemble can be advantageous, as it allows the capture of both timing inefficiencies and structural risk factors. From an operational perspective, monitoring should focus not only on maximum and relative step durations but also on retry frequency and process structures that introduce branching. This dual focus ensures early detection of both gradual degradations and sudden failure risks. It would be beneficial to integrate these findings into real-time dashboards, where operators can track "early warning indicators" such as excessive step duration variance or rising retry counts, enabling preventive interventions before full flow failure occurs.

Overall, it can be concluded that the overlap between the two models' top 10 most important features is very strong. Out of 10 features each, 9 are shared between Random Forest and XGBoost. The only unique feature in Random Forest's top 10 is TotalDurationSec, while the only unique feature in XGBoost's top 10 is HasParallelBranch.

This means that the two models are largely consistent in identifying the core drivers of flow failure: bottlenecks, step duration measures, retries, and SLA

deviations. The small differences highlight their modelling biases: Random Forest favors a cumulative measure of overall runtime, while XGBoost brings in a structural feature (parallel branching) as a failure risk factor.

In conclusion, the overlap confirms the robustness of the identified predictors, while the unique features suggest complementary perspectives that can enrich monitoring. A practical recommendation is to track both total duration and process structure complexity in addition to the common features, ensuring a more comprehensive risk detection system.

Conclusion

This study demonstrates the applicability of machine learning methods to the prediction of RPA process failures using execution logs from Microsoft Power Automate. The comparative evaluation of Random Forest, XGBoost, Stacked, and Voting ensembles reveals that all models achieve high predictive performance, with ROC-AUC values above 0.93 and PR-AUC values above 0.86. Among them, the Stacked ensemble provides the best sensitivity and PR-AUC, making it particularly suited for high-risk environments where minimizing missed failures is critical, whereas Random Forest offers the highest specificity and F1 score, providing a balanced alternative with fewer false alarms. The Voting ensemble further enhances robustness across evaluation metrics, while XGBoost, although not a leader in any single category, contributes important complementary insights, particularly regarding structural complexity of the flows.

The analysis of feature importances highlights that both Random Forest and XGBoost strongly converge on the same set of predictors, with nine out of ten features shared across their top rankings. These include step duration metrics, bottleneck indicators, retries, and SLA deviation, confirming that process timing and execution inefficiencies are central drivers of failures. The unique emphasis of Random Forest on total execution time and of XGBoost on parallel branching indicates that combining these perspectives yields a more comprehensive view of operational risks. The findings underscore the value of a hybrid monitoring strategy that integrates both timing-based and structure-based signals to provide early warning indicators for RPA process instability.

At the same time, several limitations should be acknowledged. First, the dataset used in this study is limited to 2000 flow executions across a relatively small number of processes within a defined time window. This constrains the generalizability of the findings, as larger and more heterogeneous datasets may expose additional patterns or alter feature importance rankings. Secondly, while data balancing techniques were applied to mitigate class imbalance, oversampling strategies can introduce bias, and future research should investigate more complex methods, including cost-sensitive learning or anomaly detection frameworks. Third, the present analysis focuses on static classification; however, in real-

world deployments, process behavior evolves dynamically, and models must adapt to changing environments. Finally, the study evaluates only a limited set of classification algorithms, excluding deep learning and sequence-based approaches that may capture temporal dependencies in execution logs more effectively.

Future work should therefore address these limitations by expanding the dataset to cover more processes, longer observation horizons, and diverse organizational contexts. Research should also explore advanced feature engineering techniques, particularly those capturing temporal dynamics, resource utilization, and inter-process dependencies. Hybrid modelling strategies that combine ensemble learning with deep sequence models such as LSTMs or Transformers may further improve predictive performance. The integration of predictive models into real-time monitoring dashboards and automated remediation pipelines is an important direction to transform prediction into actionable operational insight.

In summary, the proposed framework demonstrates that machine learning models trained on execution logs can serve as effective predictive tools for RPA failure detection. By identifying both gradual process inefficiencies and acute structural risks, such models provide organizations with early warning capabilities that enhance process reliability, reduce downtime, and strengthen business continuity.

Acknowledgement

This research is funded by scientific project PNI-KS24-03-ISAUPM.

REFERENCES

Antonova, D., Beloeva, S., & Todorova, A. (2024). The dual impact of artificialiIntelligence: Catalyst for innovation or threat to stability. *Strategies for Policy in Science and Education, 32*(6s), 49 – 58. https://doi.org/10.53656/str2024-6s-4-dua.

Biolcheva, P., & Shterev, N. (2024). A model for calculating the indirect added value of AI for business. *Strategies for Policy in Science and Education, 32*(3s), 9 – 17. https://doi.org/10.53656/str2024-3s-1-mod.

Canciglierie, A. B, Rocha, T., Szejka, A. L., Coelho, L. S., & Canciglieri, O. J. C. (2021). Real-time machine learning automation applied to failure prediction in automakers supplier manufacturing system. *IFIP International Conference on Advances in Production Management Systems (APMS), 2021, Nantes, France*, 303 – 310. https://inria.hal.science/hal-04030342v1.

Hall, T., & Rasheed, K. (2025). A survey of machine learning methods for time series prediction. *Applied Sciences, 15*(11), 5957. https://doi.org/10.3390/app15115957.

Iosif, A., Maican, E., Biris, S. S., & Vladut, N.-V. (2025). Application of machine learning models for predictive maintenance of biotechnical systems. *INMATEH Agricultural Engineering*, *75*(1), 930 - 949. https://doi.org/10.35633/inmateh-75-79.

Karuppusamy, P. (2021). Machine learning approach to predictive maintenance in manufacturing industry - A comparative study. *Journal of Soft Computing Paradigm*, *2*(4), 246 – 255. https://doi.org/10.36548/jscp.2020.4.006.

Leukel, J., González, J., & Riekert, M. (2021). Adoption of machine learning technology for failure prediction in industrial maintenance: A systematic review. *Journal of Manufacturing Systems*, *61*, 87 – 96. https://doi.org/10.1016/j.jmsy.2021.08.012.

Majid, H., & Anuar, S. (2022). A systematic literature review of failure prediction in production environment using machine learning technique. *International Journal of Innovative Computing*, *12*(1), 81 – 90. https://doi.org/10.11113/ijic.v12n1.348.

Mallreddy, S. R., & Vasa, Y. (2024). Predictive maintenance in cloud computing and devops: ML models for anticipating and preventing system failures. *Natural Volatiles and Essential Oils, 10*(1), 213 – 219.

Minkov, I., & Ivanov, Y., (2024). Impact of corporate culture publicized on the Internet on the organizational performance of insurance companies in Bulgaria. *Industrial Growth Conference 2024*, 366 – 375. https://doi.org/10.53656/igc-2024.31.

Ojeda, J. C. O., Borsato de Moraes, J. G., de Sousa Filho, C. V., de Sousa Pereira, M., de Queiroz Pereira, J. V., Dias, I., Monteiro da Silva, E. C., Peixoto, M. G. M., &. Gonçalves, M. C. (2025). Application of a predictive model to reduce unplanned downtime in automotive industry production processes: A sustainability perspective. *Sustainability, 17*(9), 3926. https://doi.org/10.3390/su17093926.

Patrício, L., Varela, L., & Silveira, Z. (2025). Proposal for a sustainable model for integrating robotic process automation and machine learning in failure prediction and operational efficiency in predictive maintenance. *Applied Sciences*, *15*(2), 854. https://doi.org/10.3390/app15020854.

Tengku Asmawi, T. N., Ismail, A. & Shen, J. (2022). Cloud failure prediction based on traditional machine learning and deep learning. *Journal of Cloud Computing, 11*, 47.

Wewerka, J., & Reichert,, M. (2020). Towards quantifying the effects of robotic process automation. *IEEE 24th International Enterprise Distributed Object Computing Workshop, Eindhoven, Netherlands*, 11 – 19. https://doi.org/10.1109/EDOCW49879.2020.00015.

✉ **Assoc. Prof. Yanka Aleksandrova, PhD**
University of Economics – Varna
77, Kniaz Boris I Blvd.
9002 Varna, Bulgaria
E-mail: yalexandrova@ue-varna.bg

✉ **Assist. Prof. Mihail Radev, PhD**
University of Economics – Varna
77. Kniaz Boris I Blvd.
9002 Varna, Bulgaria
E-mail: radev@ue-varna.bg