Chair of Computing in Civil and Building Engineering
TUM School of Engineering and Design
Technical University of Munich

Festo Plant for Discrete Manufacturing

Developing Simulation Models for Industrial Copilot

Submitted by
Eduardo Silva (03805057), Nien-Ying Lin (03803153), Adrián Almohalla (03804037)

Supervised by: Mohamed Khalil
Submitted on: January 18, 2026

Contents

1 Introduction

As part of the lecture "Software Lab 2025" class, this report was designed to showcase the work done in the project. Such project was managed by the supervisor Mohamed Khalil from Siemens Technology AG and the course coordinators at TUM.

This work presents the development of an intelligent software solution aimed at supporting decision-making in industrial drilling operations carried out in a Festo manufacturing environment. The project concentrates on the design and implementation of data-driven systems capable of analyzing process conditions and providing meaningful predictions related to tool behavior and machining outcomes.

By the usage of a dataset, tools and auxiliary functions were developed to be used by the industrial copilot. The main purpose of the system is to investigate the use of agent-based architectures, considering both single-agent and coordinated multi-agent approaches, to estimate tool wear progression and drilling quality under varying process parameters. These agentic configurations enable task specialization and structured reasoning when responding to user queries.

The main objective is to enhance production efficiency and decision-making through precise simulations of forces applied on drilling tools, from which degradation and quality of the discrete manufacturing process can be deduced Heredia R., 2023. The project uses empirical models to estimate wear and quality while using LangChain to build a natural language User-Agent interface. Through literature, datasets are acquired and feature importance techniques are used to correctly estimate the desired quantities; while for the second part, complete developed tools are implemented as part of GenAI-based pilots. Lastly, analyses are done to test different Large Language Models, Prompt Engineering and Agentic Architecture.

As this project was heavily dependent on coding and scripting for the creation of such agents, it is possible to find the code, its correspondent documentation, dataset and much more under the following link: https://github.com/MiRoSi-52wab/LLMs-Drill-Wear

2 Methodology and Results

As for many related topics in the field, it is important to initially define a methodology that can be approached in order to develop the product. For this project, the following components were characterized as essential and modular enough so that work parallelization could be done:

- **Dataset**: Initially, a dataset was referred by Siemens[1] since it contained valuable information for festo plant discrete manufacturing. Nevertheless, as any other dataset, it is necessary to filter and understand better the features of the dataset so that this can be properly used.
- **Degradation Agentic Tool**: It is of interest for users to be able to determine what the degradation of the drill bit will be for certain system parameters. For this reason, it is necessary to intra and extrapolate the values of the dataset to get an accurate picture of the wear in a drilling process.
- **Drilling Quality Agentic Tool**: similarly, the users might also be concerned about the quality of the drill process. By using a similar approach as for the degradation, it is possible to determine such.
- **Single and Multi-Agentic Systems**: Lastly, after developing such agentic tools, it is necessary to implement them into an agentic system. After this is done, accuracy and efficiency of the single and multi-agents are studied with respect to different prompting techniques and Large Language Models.

Due to the group size, as well as the compartmentalization offered by the different tasks of the project, it was decided that the effort should be divided between the topics. With this in mind, both agentic tools were developed in parallel while a skeleton of agentic systems was built. After tools implementation, these could be tested inside the agentic systems, leading to independent pieces of the project that get joined together after individual development.

Now all the above topics are explored in more detail, exposing the process that led to their development as well as implementation.

2.1 Dataset

After receiving the dataset from Siemens, the group looked into it and the publication Wallsberger R., 2023 from the dataset's team gaining advantageous information about the nature of the data.

As it is seen in the dataset, there are mainly four types of failure that are evaluated for each system's configuration and parameters. Given these failures, by the usage of decision tree models (such as the XGBoost), it is possible to create decision splits and understand

[1]The dataset is not property of Siemens and it is available publicly.

which features represent a bigger weight when making a prediction for failure. Using the values for all the failure modes, the following results is obtained in the dataset's paper:

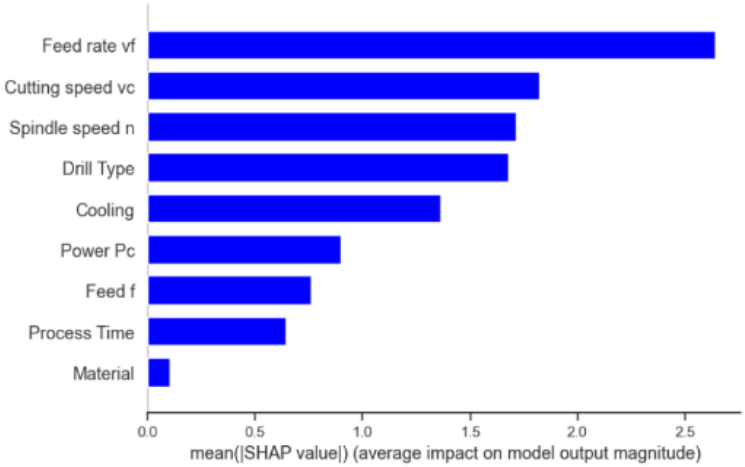

Figure 1: Feature Importance for Failure Modes.

As seen above, feed rate and cutting speed represent the highest influence on the outcome of failure in the drilling process. With this in mind, it is possible to understand that for all the failure criteria these two features are necessary to correctly predict the outcome.

However, not all failure modes are of interested for each tool. As it will be introduced in section 2.2, only the flank wear failure (FWF) is of importance to correctly predict the degradation of the drill bit. The different modes of failure are explained in the dataset's paper and below:

Build-up edge failure (BEF): Represented as a binary feature, a build-up edge failure indicates the occurrence of material accumulation on the cutting edge of the drill bit due to a combination of low cutting speeds and insufficient cooling. A value of 1 signifies the presence of this failure mode, while 0 denotes its absence.

Compression chips failure (CCF): This binary feature captures the formation of compressed chips during drilling, resulting from the factors of high feed rate, inadequate cooling, and the use of an incompatible drill bit. A value of 1 indicates the occurrence of at least two of the three factors above, while 0 suggests a smooth drilling operation without compression chips.

Flank wear failure (FWF): A binary feature representing the wear of the drill bit's flank due to a combination of high feed rates and low cutting speeds. A value of 1 indicates significant flank wear, affecting the drilling operation's accuracy and efficiency, while 0 denotes a wear-free operation.

Wrong drill bit failure (WDBF): As a binary feature, this indicates the use of an inappropriate drill bit for the material being drilled. A value of 1 signifies a mismatch, leading to potential drilling issues, while 0 indicates correct drill bit usage.

Now it is possible to repeat the feature importance decision tree model for each specific type of failure. With this, it is possible to understand which parameters most affect which different failure. As an example, the paper also shows this feature importance for FWF (later used for the degradation tool):

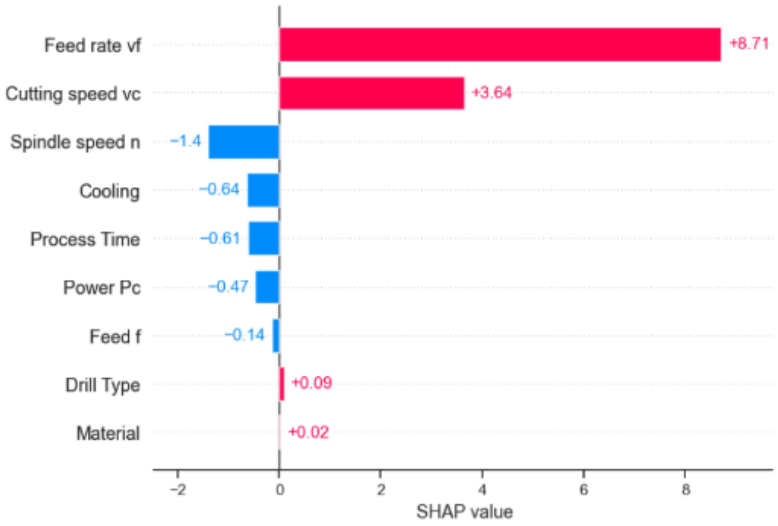

Figure 2: Influence of Features for Flank Wear Failure Prediction.

As seen above, once again feed rate and cutting speed are the parameters that most affect the probability of flank wear failure occurring. It is however interesting to notice that the processing time is not of importance. This is caused since all the processing times in the dataset are around $\simeq 30$ sec, leading to a very monotone testing scenario. Nevertheless, it is known that for drilling processes, this is usually the time scale used, leading to a consistent analysis and study. This way, while being constrained by the fact that the processing time should always be around 30 sec for good predictions, it also reduced the complexity of the system's response leading to better predictions.

2.2 Degradation tool

First of all, the data from Siemens dataset was used to create different graphs and see which of the parameters were suitable to create a regression model which tell you the tool degradation.

Tool degradation state was divided into three classes:

	FWF (Flank Failure)	Main Failure (any type of failiure)
Class 2 (worn)	True	True
Class 1 (moderated wear)	False	True
Class 0 (low wear)	False	False

Table 1: Degradation classification

After that, parameters which complies the regression model where set as vc (spindle speed) and Tf (Torque/ feed) for each three drilling bit materials, where torque was calculated as follows:

$$T\,[\mathrm{Nm}] = \frac{9550 \cdot P_c\,[\mathrm{kW}]}{n\,[\mathrm{min}^{-1}]} \tag{1}$$

, where 9550 its an unit conversion factor that ensures everything is in SI units.

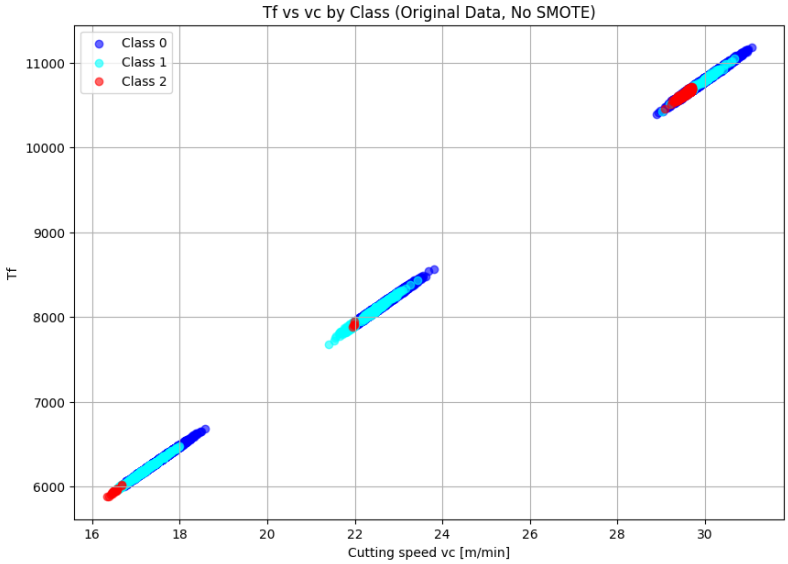

Figure 3: Classification of degradation model

From Figure 3, it was observed that the three different classes where mixed so a filter, setting max values per class, was applied to have this solved:

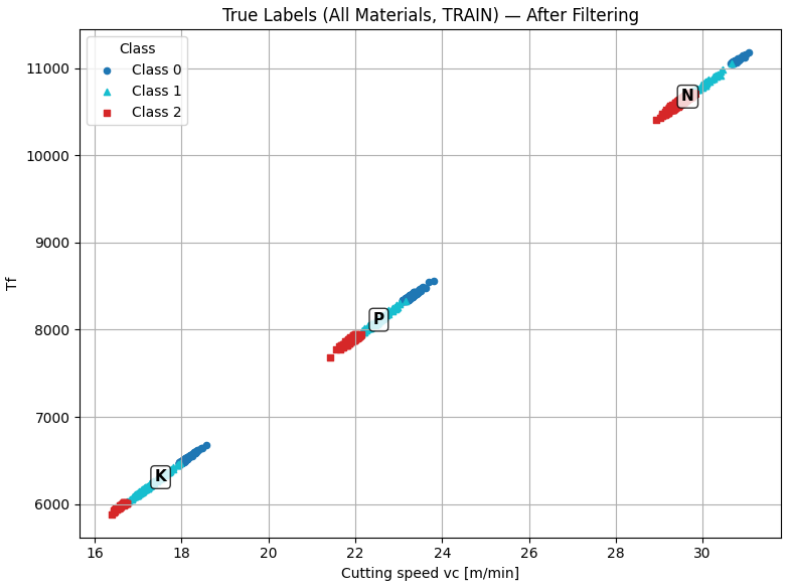

Figure 4: Classification after filter

With the data properly structured, a machine learning model was trained using XGBoost to predict tool wear conditions. Class imbalance was handled through sample weighting, and cross-validation with early stopping was applied to prevent overfitting. The final classifier outputs probabilistic predictions for each wear zone that will be used for the wear predictionErtnunc H., 2004.

Lastly, a display was generated for predicting wear with new inputs, where user choose the material being drilled (P, N or K)[2], feed and vc and gets the predicted wear to know if he should change the drilling bit or not as showed in Figure 5.

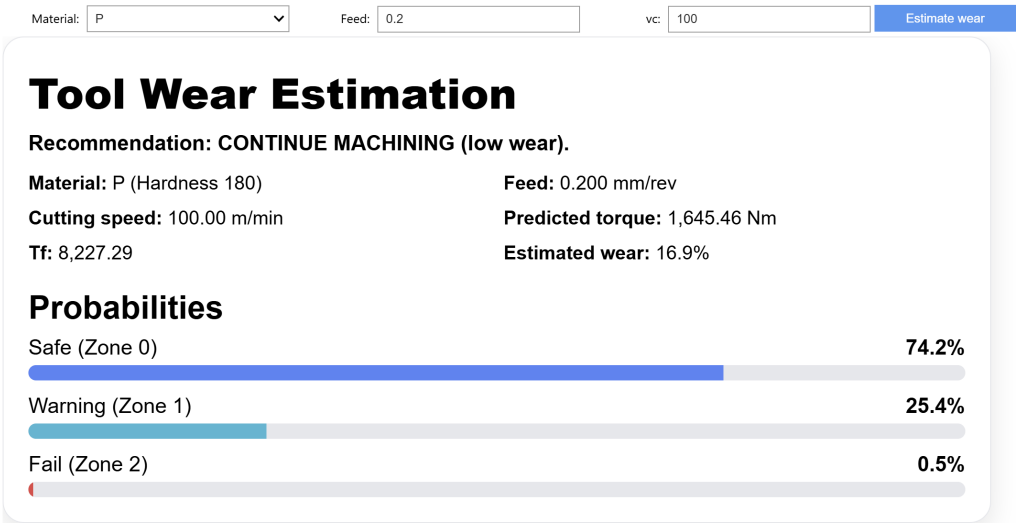

Figure 5: User interface of the degradation tool

[2]The three materials are represented as “P (Steel)” for C45K, “K (Cast Iron)” for cast iron GJL and “N (Non-ferrous metal)” for AlSi alloy.

For this last step, the probabilistic predictions from the classifier are used to estimate the wear following the next equation:

$$\mathrm{Wear}[\%] = 5\,p_0 + 50\,p_1 + 100\,p_2. \tag{2}$$

, where each "p" refers to the probability of being in class 0, 1 or 2. The multipliers used in equation 2 are estimations based on the next graph from the paper that introduces the dataset, where class 0 is zone 1 and 2, class 1 zone 3 and 4 and class 2 zone 5:

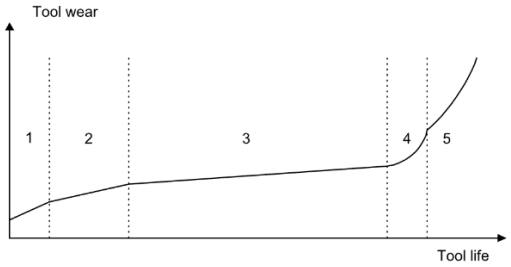

Figure 6: Failure graph from dataset experiments

To assess the performance of the predictive model, it was reported a confusion matrix and a performance metrics comparison between classes.

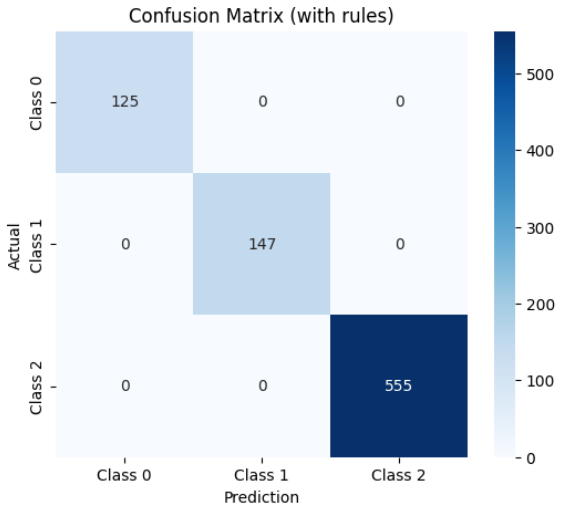

Figure 7: Confusion matrix of degradation tool

From Figure 7, its observed that the model has a perfect recall and precision due to the maximums per class that were setted (without them model fails to identifying 3 points as class 0 instead of class 1).

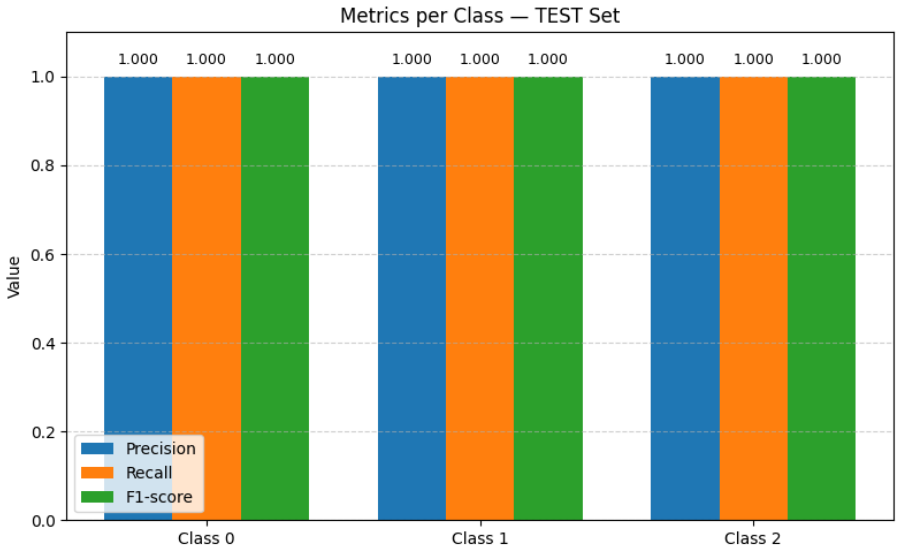

Figure 8: Performance metrics of test set

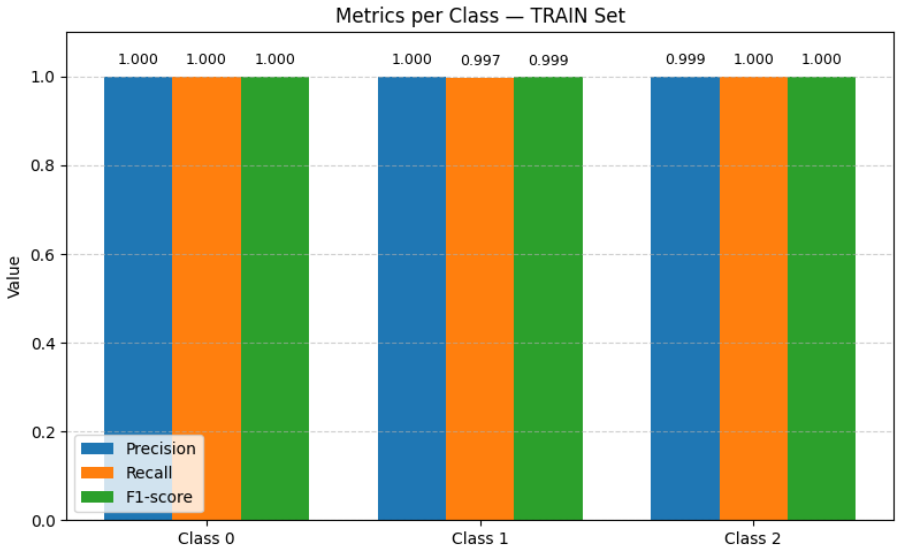

Figure 9: Performance metrics of train set

Finally, from Figures 8 & 9 its seen as with the confusion matrix that precision and recall are perfect. Apart from that, its noticed that F1-score is 0.999 for training set while 1 for the test set, and this is good since smaller F1 value for training than testing data in cross validation)[3] means that model is generalizing good and not memorising everything.

The data used for the machine learning model was filtered and due to that, the precision and recall from the classification model in the train and validation part are almost 1.00.

[3]Technique that was used to avoid overfitting.

2.3 Quality tool

After reviewing several studies on drilling quality, it is evident that most existing approaches rely on **experiment-specific data**. Drilling behavior is highly sensitive to machining parameters, tool geometry, and small variations in workpiece material composition, particularly in metallic alloys, which significantly limits the generalizability of quality models.

Since additional drilling experiments were not feasible within the scope of this project, the quality-related component is instead based on the available **failure labels** in the dataset, which are used as a proxy for drilling quality. In this part of the project, we focus on two failure labels provided in the dataset:

- **Build-up Edge Failure (BEF)**: BEF refers to the formation of built-up material on the cutting edge, which alters the effective tool geometry and often leads to poor surface finish and unstable cutting conditions.
- **Compression Chips Failure (CCF)**: CCF is associated with unfavorable chip formation, where chips become excessively compressed or accumulated, resulting in impaired chip evacuation and degradation of drilling quality.

Together, BEF and CCF represent two common and practically relevant failure modes related to drilling quality.

Implementation

Model Setup and Training The quality tool is implemented using a **decision tree classifier**, selected for its interpretability and suitability for explicit rule extraction. Separate models are trained for BEF and CCF as independent binary classification tasks. The input features consist of all available process-related variables in the dataset excluding failure labels, while the output indicates whether the corresponding failure occurs. From a quality monitoring perspective, **recall is prioritized** over precision, as failing to detect a potential failure is considered more critical than issuing conservative warnings Lee Y., 2023.

Split-Seed Analysis The dataset is **highly imbalanced**, which leads to noticeable variations in recall–precision trade-offs across different train–test split seeds, as illustrated in Fig. 10. In particular, models trained with different random splits may exhibit similar overall performance while achieving substantially different recall levels. To assess model robustness and to avoid conclusions based on a single data split, multiple split seeds are evaluated during training.

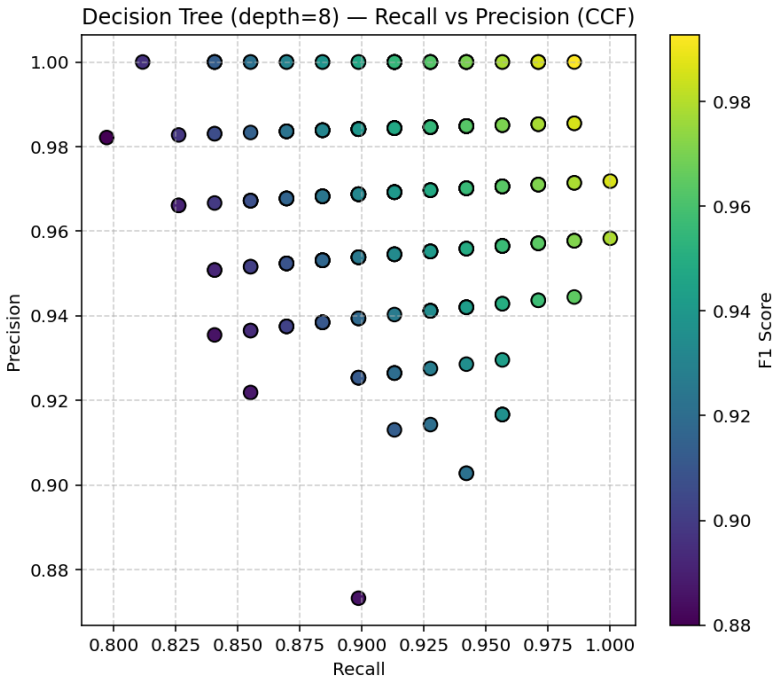

Figure 10: Recall–precision distribution across different train–test split seeds

Rule Extraction and Interpretation After training, failure-related decision paths are extracted from the decision trees and converted into human-readable rules for downstream use. To ensure interpretability and physical plausibility, model behavior is analyzed from multiple perspectives, including decision tree rule extractions, feature importance, and confusion matrices. A simplified decision tree visualization for BEF (Fig. 11) is included to illustrate the main decision paths used for rule extraction, providing an interpretable overview of how failure conditions are derived from the learned model.

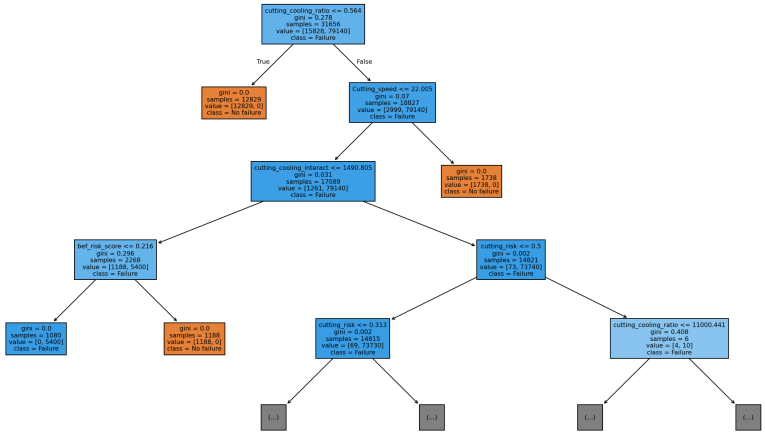

Figure 11: Simplified decision tree structure for BEF

The number of extracted **BEF rules are four**, which is well suited for downstream use by the agent without additional post-processing. In contrast, further refinement is required for CCF due to its higher initial rule complexity.

To assess the predictive performance of the extracted BEF rules, a confusion matrix is reported in Fig. 12. The results indicate perfect classification performance on the test set, with both precision and recall reaching 1.00, demonstrating that BEF-related failure conditions can be reliably captured by a small and interpretable rule set.

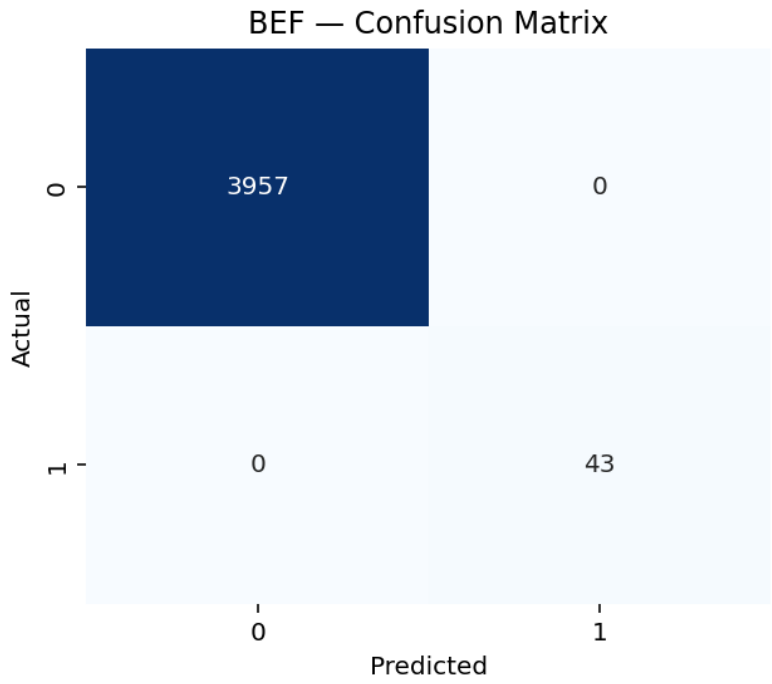

Figure 12: Confusion matrix for BEF (Precision = 1.00, Recall = 1.00)

The corresponding feature importance analysis for BEF and CCF is shown in Fig. 13 and Fig. 14. For BEF, **cooling and cutting-related parameters** dominate the decision process. This is directly aligned with the dataset's generation logic, where BEF is defined by a combination of low cutting speeds and insufficient cooling.

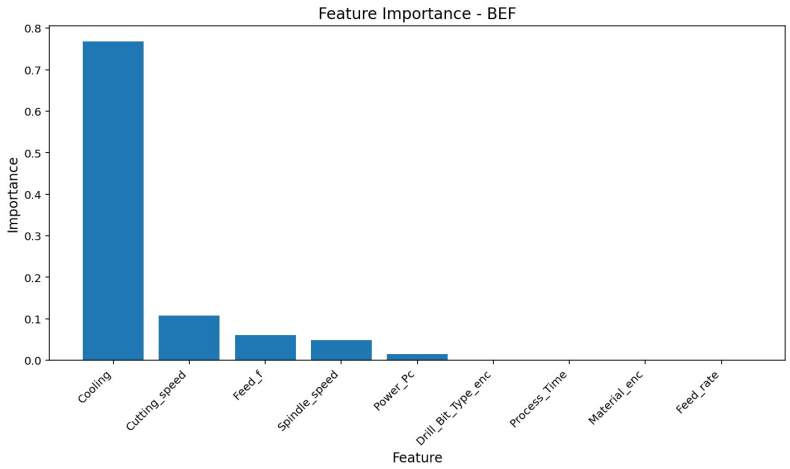

Figure 13: Feature importance for BEF

For CCF, **feed rate, cooling, and drill bit type** emerge as the most influential features. These results accurately reflect the underlying failure mechanism of the dataset, which

triggers a CCF label based on high feed rate, inadequate cooling, and the use of incompatible drill bits. Overall, the feature importance results confirm that the decision tree successfully captured the predefined physical rules governing the synthetic dataset.

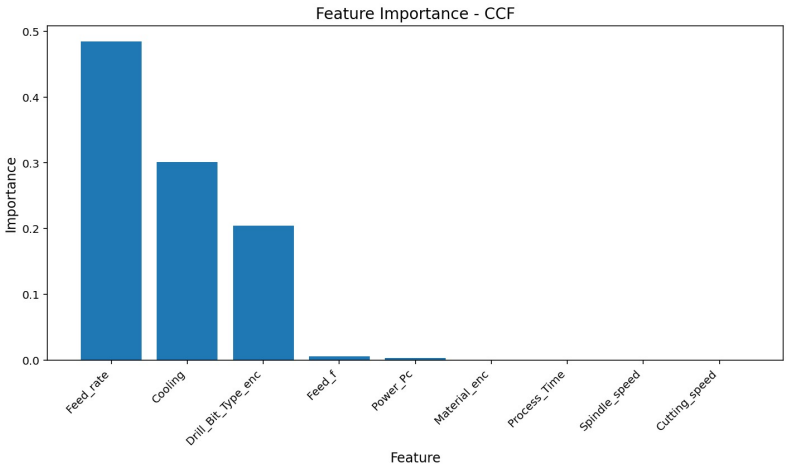

Figure 14: Feature importance for CCF

Rule Reduction for CCF The initial decision tree for CCF produced 18 failure rules, a level of complexity that increases the risk of overfitting and reduces human-interpretability. To streamline the model, two progressive post-processing strategies were explored:

1. **Rule Merging**: By combining similar or overlapping decision paths, the rule set was first reduced **from 18 to 8**. This process successfully maintained a perfect **recall of 1.00** with only a marginal decrease in precision (0.96), as shown in Fig. 15. This stage demonstrates that redundant complexity can be eliminated without missing any critical failure cases.

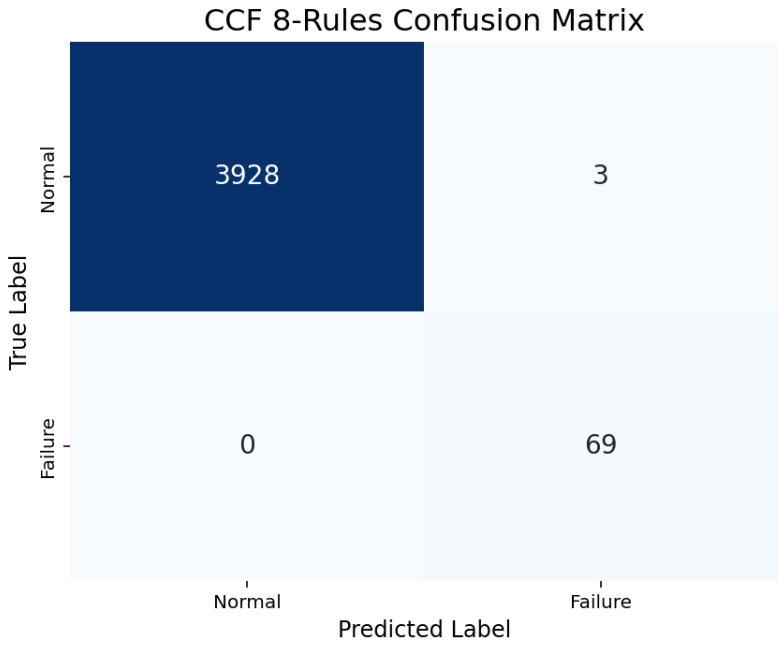

Figure 15: Confusion matrix for CCF with 8 rules (Precision = 0.96, Recall = 1.00)

2. **Rule Elimination**: To further enhance system deployability, rule elimination was applied to reach a compact set of **4 rules**. As illustrated in Fig. 16, this additional reduction involves a strategic trade-off: it results in a small decrease in recall (from 1.00 to 0.94), meaning a few failure cases are no longer detected in exchange for a significantly more interpretability-focused rule base.

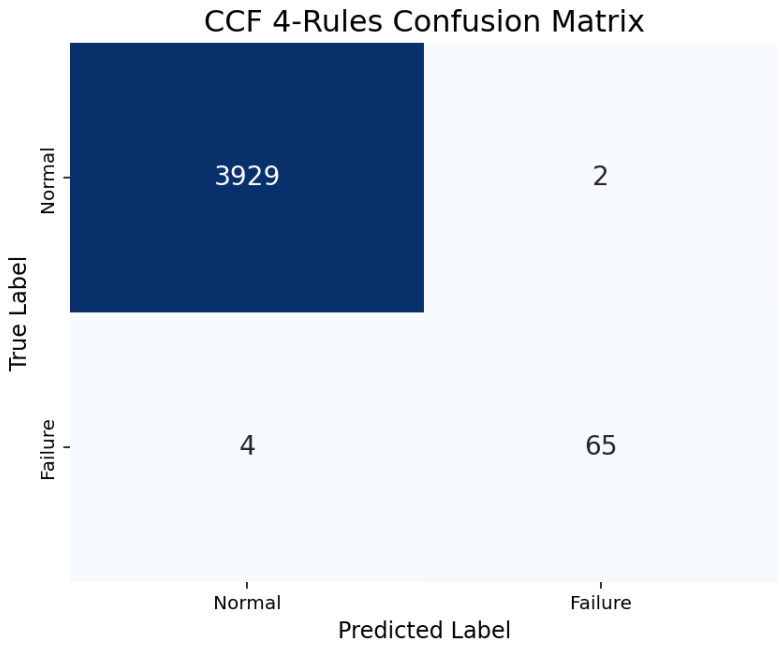

Figure 16: Confusion matrix for CCF with 4 rules (Precision = 0.97, Recall = 0.94)

These results highlight that while merging is an effective way to remove redundancy, reaching extreme compactness necessitates a conscious trade-off between simplicity and full failure coverage.

Overall, the quality tool demonstrates that **interpretable**, **rule-based** models can effectively capture drilling failures using the available failure labels. BEF can be represented with a compact rule set, whereas CCF requires a more nuanced reduction process. The results indicate that CCF-related information is distributed across multiple process variables, making rule-level simplification more effective than simple feature restriction. In practice, maintaining full failure coverage imposes a constraint on how compact the final rule set can be, highlighting the necessary **trade-off between simplicity and recall** in quality monitoring.

2.4 Agentic Systems

While developing the tools, it is necessary to build the skeletons for the agentic systems, allowing for the quick implementation and testing of such tools. As referred initially in the report, it is of interest to evaluate two different architectures for the industrial copilot:

- **Single Agent System**: This system is characterized by only containing one agent. The agent is responsible for all tasks that are necessary to answer the user's query (analyzing the query, using the tools and retrieving the values). Single agent systems are characterized by their easy implementation and concentration of tasks in only 1 trained entity.
- **Multi-Agent System**: For the other scenario, many agents can be coupled together to form a more complex model, where each query is processed by many agents through a logical chain. For this reason, it can be thought as many different single agent systems working on specific parts of the task and forming a response given all their inputs. ss

To better visualize both scenarios, the following example is given:

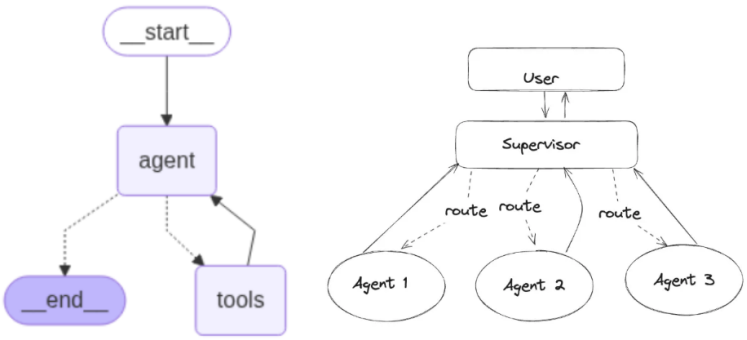

Figure 17: Single and Multi-Agent Systems.

On the left side it is possible to see an example of a single agent system. The agent is connected to the tools and communicates with these to retrieve values based on the user's query parameters. On the other side, there is an example of a multi-agent system (with the same structure as the one used in this project), where a supervisor agent is connected to tool specialist agents. This means that the tool agents are responsible for correctly inputting and retrieving values from the tools, while the supervisor is responsible for understanding the user query and retrieve an informed answer.

For this project, it is interesting to evaluate the performance (by evaluating accuracy and efficiency) of both design models for the industrial copilot. Before evaluating the performance, some decisions need to be made regarding both architectures. In the scope of the project, it is important to carefully select which **large language mode** should be used for the agents, as well as, which **prompting technique** leads to better and quicker responses.

Large Language Model Choice

One of the key components of how an agentic system behaves is the selection of the large language model to support this agent. For this selection, given the limitations on memory usage and licenses, the Ollama library was used to locally host large language models. With these, the following loop was executed:

1. Create "neutral" prompt for LLMs.
2. Create a batch of user queries by difficulties.
3. Select a large language model.
4. Retrieve answers and response time.
5. Repeat from point 3 until all LLMs are evaluated.

For the first point in the process, it is needed to create a "neutral" prompt, such that it can be equally used for all tested LLMs. By "neutral" one can imagine a prompt that is not specific to tasks and gives the LLM freedom to create its own logical chain. This prompt should only give the minimal necessary information such that LLM knows what tools are available for its response.

By using the same prompt for every LLM, a consistent pipeline is created. It is know that this prompt will create some positive or negative bias (depending on how the LLM was trained) but it is considered that over all the user queries this bias will be reduced.

For the user queries, in the same way as the "neutral" prompt, a batch of different difficulty questions are created to be used for every single LLM. This allows for the study of the LLMs in diverse scenarios, where either simpler or more complex rational thoughts are required from the agentic system. For this reason, the queries were separated into the following criteria:

Easy Queries: These queries are straightforward, with correct values and units for material, feed rate, cutting speed, and cooling. No ambiguity in the input, designed to test if the system can handle perfectly structured queries.

Medium Queries: These queries can involve unit conversion when parameters are not given in SI-Units.

Hard Queries: For hard queries, the agent needs to use both tools (degradation and quality) together, as well as, deal with missing information for some of the parameters by alerting the user or assuming default values (informing the user of such decision).

With both the "neutral" prompt and the batch of user queries, it is possible to create a pipeline where the response from each LLM to every question is evaluate both in accuracy and time of response. For the first, no automatic process was developed, leaving the

evaluation criteria to be done manually by the group; whilst for the second, the response times were averaged over the queries of each difficulty. The following plot shows the results over the LLMs that were tested:

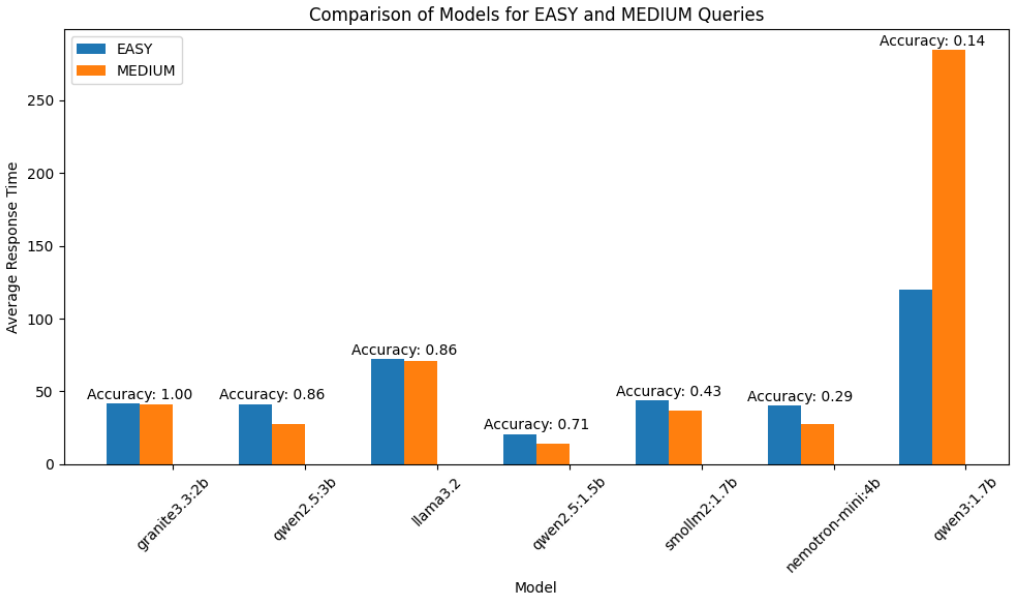

Figure 18: Performance of different LLMs.

As seen above, from the 7 different LLMs that were tested (no more could be used due to local restrictions on RAM), it is possible to infer that **granite3.3:2b** gave the most accurate response while still being fast in doing so. With this in mind, this was the selected LLM to be used in both the single and multi-agent systems.

Given the LLM to be used, it is now important to refine the "neutral" prompt for both agentic systems and evaluate them again over the user queries. After this, it will be possible to create a clear view of the performance of both systems and compare them.

Prompt Engineering for Agentic Systems

For both copilot architectures (single and multi-agent), it is important to understand the different types of prompting techniques, as well as what the prompt's role is. One can imagine the prompt to be the guideline that tells the LLMs how to behave when queried, similar to giving instructions on what the expected input and output should be.

Many different prompting techniques have been developed, and the literature on this topic is vast. For that reason, only some of the most common ones were used in the scope of this project, to create a simpler analysis of all the possible techniques. For both agentic systems, prompts were created following:

- **Zero-Shot Prompts:** Zero-shot prompting consists of providing the model with a task description without any examples of the expected output. The model is required to rely solely on its pre-trained knowledge and generalization capabilities to infer the

correct response. This technique is useful for evaluating the baseline reasoning and adaptability of the model.

- **Few-Shot Prompts:** Few-shot prompting extends zero-shot prompting by including a small number of input–output examples within the prompt. These examples serve as implicit guidance, helping the model recognize patterns and better align its responses with the desired format or reasoning process. This approach often improves performance on structured or specialized tasks.
- **Chain-of-Thought (CoT) Prompts:** Chain-of-Thought prompting encourages the model to explicitly generate intermediate reasoning steps before producing a final answer. By making the reasoning process visible, this technique improves performance on tasks that require multi-step logical inference or numerical reasoning. It also enhances the of the model's decisions.
- **Plan and Solve (P&S) Prompts:** Plan and Solve prompting separates the reasoning process into two distinct stages: planning and execution. In the first stage, the model outlines a structured plan to solve the problem, and in the second stage, it follows this plan to generate the final solution. This technique helps reduce reasoning errors in complex tasks by enforcing a more organized problem-solving approach.

Now, by creating a batch of prompts for each technique, it is possible to evaluate what techniques work better for both architectures. It is important to know that while for the single-agent system there is only 1 prompts being used, for the multi-agent there will be three. This is because for the second scenario, we have a supervisor agent, a degradation agent and a quality agent, needing to evaluate for each agent all the techniques.

Once again, the responses from the architectures were evaluated manually as there is still no clear idea of what could be used to automate this process (could be interesting to create another LLM entity just to evaluate responses), while the time processes were averaged for query difficulty in the single agent case, and for type of agent for the multi-agent system. This lead to the following plot for the single agent system:

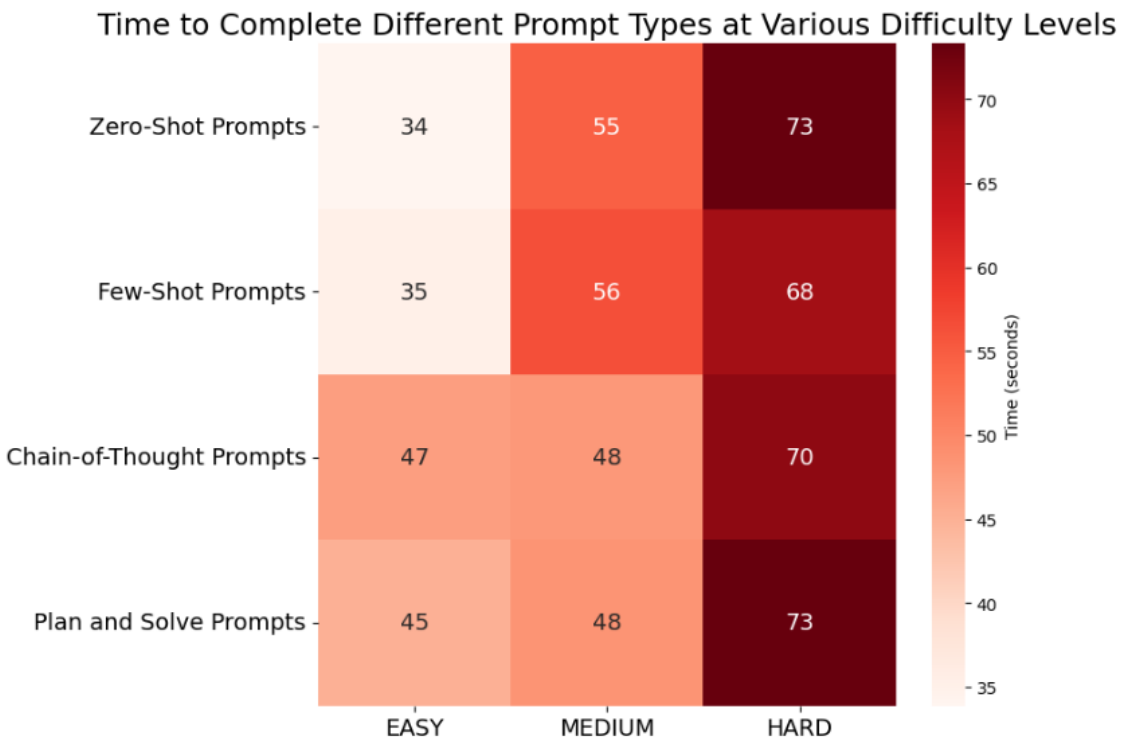

Figure 19: Prompt Engineering for Single Agent System.

As seen above, for the single agent system, both zero-shot and few-shot techniques increase their response time drastically proportionally to the query's difficulty. This can indicate that the prompt is to simple and leads to a longer time of rational thinking by the LLM deal with the task. On the other hand, CoT and P&S techniques show a more constant behavior, indicating that the system is carefully analyzing all queries as the same. For this reason, for the single agent system, it was decided to use Chain-of-Thought Prompting technique.

For the multi-agent system, the following plot was obtained:

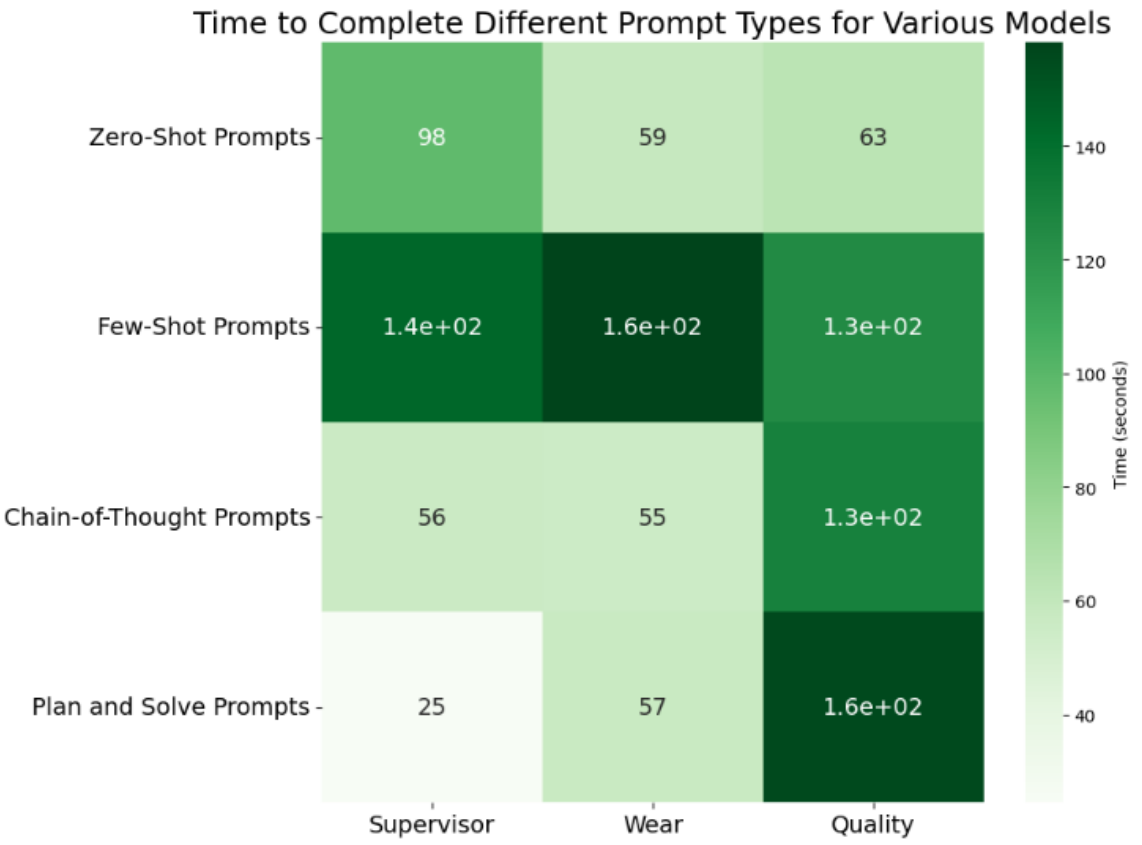

Figure 20: Prompt Engineering for Multi-Agent System.

In this case, it is necessary to evaluate the best technique for each of the agents used:

- **Supervisor Agent**: Plan and Solve shows the lowest response time, indicating that the main task of the supervisor is only to build a map of actions that need to be done in order to answer the query. This is expected, as the supervisor should not be concerned with using the tools.
- **Wear and Quality Agents**: For these two, it is possible to see that zero-shot prompts are quicker. This makes sense, as the only purpose for this agents is to use the tools developed and retrieve the correct values from such.

This combination leads to the better prompting technique of the multi-agent system. Now that both systems are implemented, in section 3 we look into comparing both architectures structured already with the best LLM (granite3.3:2b) and corresponding prompting techniques.

3 Results: Agentic Architectures Comparison

As mentioned before in the methodology section, it is now of interest to finally compare both agentic structures. By using the best LLM (granite3.3:2b) and the corresponding best prompting techniques for each system, it is possible to once again evaluate accuracy and response time on the user queries. By doing this, the following plots are obtained:

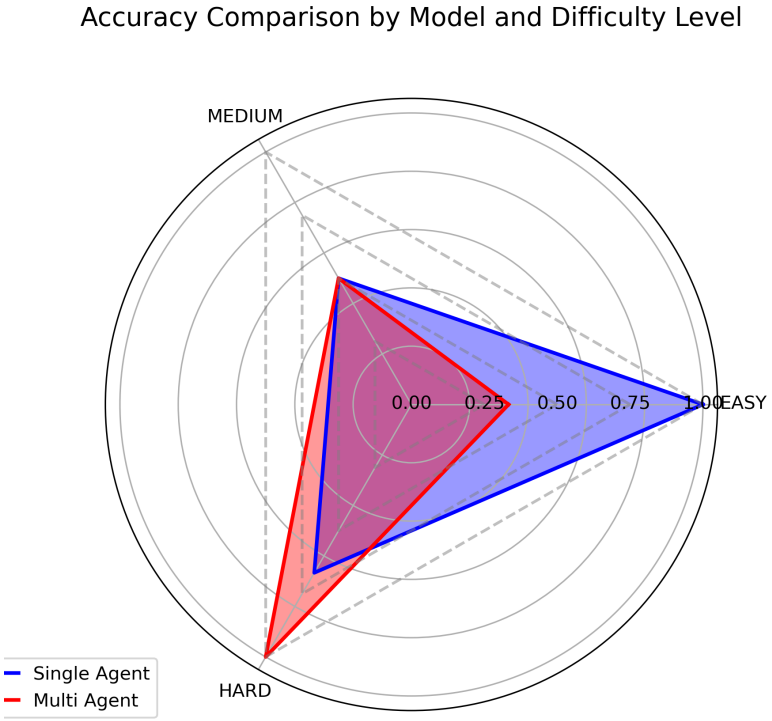

Figure 21: Accuracy for Single and Multi-Agent System.

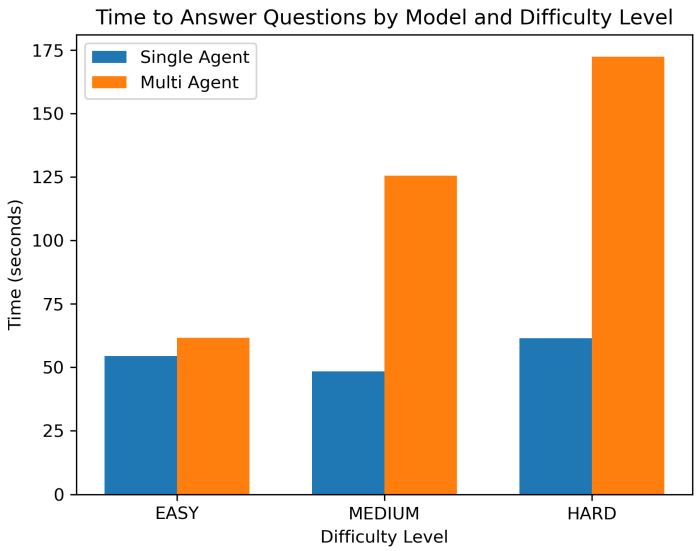

Figure 22: Response Time for Single and Multi-Agent Systems.

By looking at both plots, it is possible to understand the bigger picture of the agentic systems. While the single agent outperforms in response time, being able to respond quickly to any question, its accuracy is only acceptable for easy questions. On the other hand, the multi-agent perfectly deals with complex queries that require more compute, but takes toll by taking longer to fully rationally understand it and respond correctly.

Given the constraints of the project (RAM usage, compute power, etc...) which agentic structure should be used? The answer to this questions depends solely on the type of application that the industrial copilot is deployed. When the goal is to attract new users to learn about the drilling process in a festo plant, accuracy might not be a priority for the application. However, in a setting where it is necessary to maintain quality of drilling and precision at high standards, it is of most importance to get correct values.

Nevertheless, it is possible to understand with this project the advantages and constraints of both architectures, leading to a better informed decision when needed to decide between both.

4 Future work

Regardless of all the implementations made for this project, there is always hope that new tools can be developed and implemented into the agentic system. Additionally, improvements to the already implemented tools can lead to new scenarios, for example: allowing the tools to predict wear and quality for new materials that are not present in the dataset, or also quantify in a continuous matter the quality of drilling (not relying only in "Good" or "Bad" criteria).

On the agentic system's side, it is possible to evaluate new LLMs that are not restrained by RAM. Other approach is to increase the number of user queries and subdivide each level by topics, leading to a more extensive and less biased batch of queries for evaluation of the system. Additionally, as already introduced before, it would be interesting to create another LLM system that is responsible for evaluating the agentic system's responses, making it not necessary for this part to be done manually.

Lastly, more prompt techniques can be used and tested for both architectures, as well as more diverse architectures. Creating more possible frameworks and pipelines can lead to a more extensive work, creating a better view of what best fits the demands of the application of the industrial copilot.

5 Conclusion

This project developed and evaluated an intelligent “industrial copilot” prototype to support decision-making in drilling operations in a discrete manufacturing setting. Building on a publicly available synthetic failure-mode dataset, two data-driven tools were implemented: (i) a degradation tool that estimates drill wear progression, and (ii) a drilling quality tool that predicts quality-relevant failures using interpretable, rule-based logic. These tools were then integrated into two agentic architectures—a single-agent system and a coordinated multi-agent system—and bench marked under different large language models and prompt engineering strategies.

On the tool level, the degradation pipeline combined feature selection, class definition, and an XGBoost-based classifier to produce probabilistic wear-zone predictions, which were converted into a continuous wear estimate for user-facing decision support. The quality component prioritized interpretability and operational safety by using decision trees and explicit rule extraction for Build-up Edge Failure and Compression Chips Failure, and it demonstrated how rule merging/elimination can trade off compactness against recall for deployability. On the agentic system level, a structured evaluation showed that the single-agent setup is consistently faster, but its accuracy degrades as query complexity increases, whereas the multi-agent setup achieves substantially better performance on complex queries by decomposing tasks across a supervisor and tool-specialist agents—at the cost of increased response time. Overall, the key outcome is that architecture choice should be driven by deployment priorities: if responsiveness and lightweight compute are dominant constraints, a single-agent assistant can be sufficient for well-formed, low-ambiguity interactions; if correctness under ambiguity and multi-step reasoning is critical for production decision support, the multi-agent approach is more suitable despite its higher latency.

Finally, the results also highlight practical limitations that bound external validity: the approach depends on dataset coverage (materials, parameter ranges, and label definitions), and performance estimates are sensitive to evaluation design and the representativeness of the query set. Within these constraints, the project demonstrates a complete and modular pipeline—from data understanding and interpretable modeling to tool integration and agentic benchmarking—that can be extended toward more realistic datasets, continuous quality metrics, and automated response evaluation in future work.

Bibliography

Ertnunc H., O. C. (2004). Drill wear monitoring using cutting force signals.

Heredia R., N. J., Alós I. (2023). Model-based tool condition prognosis using power consumption and scarce surface roughness measurements.

Lee Y., Y. H. (2023). Modelling of cutting tool life with power consumption using taylor's equation.

Wallsberger R., M. S., Knauer R. (2023). Explainable artificial intelligence in mechanical engineering: A synthetic dataset for comprehensive failure mode analysis.