Meeting Notes

Predictive Tool Life Model

Meeting Date: 29th August 2025

1. Overview

This document provides technical details of the **Predictive Tool Life Model** developed to estimate the **Remaining Useful Life (RUL)** of cutting tools in terms of cumulative load (cum_load).
The model is designed to support ML-based tool replacement decisions in the Generative Machining Application.

2. Machine Learning Objective

2.1 What is Being Predicted (and Why)

The model predicts **Remaining Useful Life (RUL)** of a cutting tool in **terms of the percentage of the remaining cumulative load**.

- **Objective**:
  - Provide real-time predictions into how much usable life remains for the current tool, and
  - translate them into user-specific insights like predicted ppe.
- **Why**: Helps avoid both premature replacements (increasing cost) and late replacements (causing tool failure or scrap).

2.2 Value Over Existing Methods

Category	Analytical Method	ML-Based Method	Value Added
Prediction Quality – Wear Curve	Assumes linear wear	Learns non-linear wear patterns	More holistic PPE estimates

Prediction Quality - Context Awareness	Uses only utilization counts	Incorporates load, feed, speed dynamics	Adjusts predictions based on actual operating conditions
Prediction Quality - Accuracy	Plateaued at rules-of-thumb	Improves with more data	Reduces premature tool changes
Explainability	Fully transparent	Requires SHAP/LIME	Harder to explain predictions in terms of ML models
Implementation Cost	Minimal	Moderate (infra + inferences + retraining)	Costlier than Analytical method

3. Model Description

3.1 Input Categories

- **Base Features:**
  - Load
  - Feedrate
  - Spindle Speed
- **Recency Features (Temporal Statistics):**
  - Rolling averages (5s, 10s, 25s, 50s)
  - Rolling standard deviations (5s, 10s, 25s, 50s)
  - Rolling maximums (5s, 10s)
- **Contextual Metadata (not passed to the model):**
  - Machine, Program, Tool ID
  - PPE thresholds (operator-set and actual replacement point)

3.2 Model Type

The predictive pipeline is an **ensemble regression model:**

- **Base Models:**
  - XGBoost Regressor

    - Random Forest Regressor
    - LightGBM Regressor

- **Meta-Model**:
    - RidgeCV Regression (learns to optimally combine base predictions).

This architecture captures both non-linear dynamics (via boosting/forest models) and ensemble averaging for robustness.

3.3 Training Objective

- **Output Variable**:
    - **RUL (%)** – normalized remaining life based on cumulative load.
    - Ex: Say a tool T in the training dataset reaches 2000 cumulative_load points, then this 2000 will be normalized as 100% life of this tool run.

      If for another tool run of that same tool T, the cumulative_load points reaches 2200, then this will be normalized to 100%.

      NOTE: We are predicting the *remaining* tool life %, which means the model starts off with 100% life remaining and ends up at 0% remaining life.

    - This normalization accounts for the dynamic nature of tool wear, since the maximum cumulative load a tool can endure varies across runs depending on machining parameters and operating conditions.
- **Primary Metric**: RMSE (Root Mean Square Error)
- **Secondary Metric**: weighted RMSE, reliability score

3.4 Model Experiments

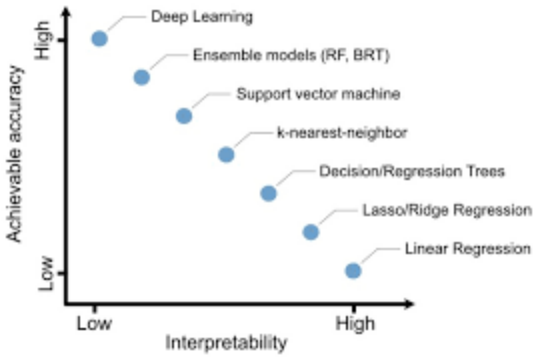
NOTE: This section dives deeper into the technical details and reasons of the experiments performed in finalizing the Tool Life Predictive Model. This section is optional from a Business stand point.

We tested multiple approaches. The scope of these model attempts was limited to Traditional ML models instead of Deep Learning models because

1. Deep Learning models need a LOT of data

- Acquiring so much data may not be suitable or feasible for our use-case.
- If a tool's lifespan lasts a few minutes, like 2 minutes, it would have at most 150 datapoints per tool change round.
  In general, if we even try to acquire 10,000 datapoints (still less in terms of what DL models need) we may need 66 rounds of tool changes here. So, this can't be accepted.

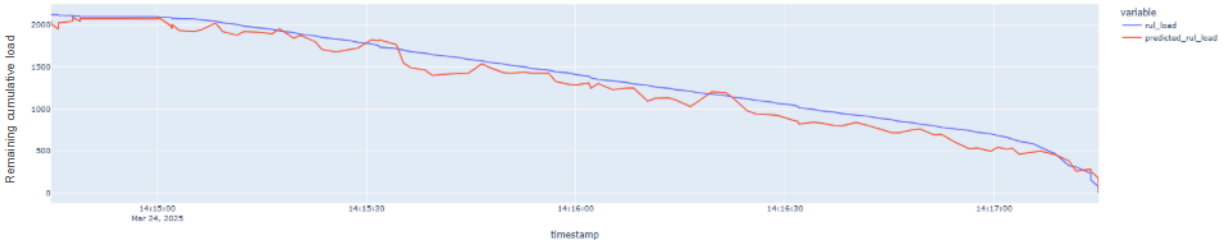
2. DL models are compute heavy.
   - We want quick predictions/computations by the model for the predictions to be relevant.
3. DL models are extremely difficult to interpret.
   - They are considered blackbox models, though we have techniques that can help give us intuition towards the predictions.
   - But when we want to decide upon a whether to use a high accuracy Deep Learning model or a good accuracy and good interpretability model, the traditional ML models are preferred.



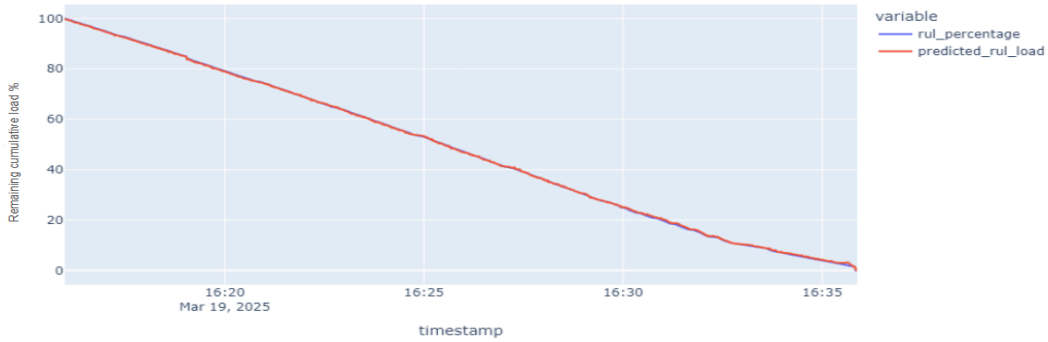
Models Tested and their results:

1. **Tree-Based Models (XGBoost, LightGBM, Random Forest):**
   - Pros: Are lightweight models, suitable for multiple model instances
   - Cons: Not robust to noise, fluctuating predictions.
   - RMSE consistently **70-80** per model
   - The graph below shows a sample tool life from the validation set for a specific Machine-Program-Tool combination. It shows that the

the model is able to understand the overall curve of tool wear but is highly fluctuated in its decisions.



2. **Survival Analysis:**
    - Pros: Added probabilistic interpretation
    - Cons: Very high error rates, need the curve to be defined by a mathematical function (infeasible for every Machine-Program-Tool combination)
    - Inconclusive results, need further analysis to define robust functions and methods for applicability
3. **Ensemble Meta-Learner (RidgeCV on top of base tree models):**
    - Combined strengths of boosting + bagging
    - Multiple Predictive Models with Meta-Level Aggregation:
        i. Individual models predict tool life based on historical data specific to each tool.
        ii. A meta-model learns the best way to aggregate these predictions, optimizing the final tool life estimate for each tool.
    - Final RMSE ~**4%** across validation sets
    - Most stable across dynamic Machine-Program-Tool combination (fig below)



Decision: We finalized the **Tree-Based Ensemble Model** because it offered:

- Best balance of accuracy, robustness, and interpretability.
- Incremental retraining possible.
- Compatibility with SHAP for feature importance.

4. Training and Inference

The model follows an **incremental learning strategy**, where it continuously improves as more data becomes available (in this case, more the tools are changed.)

4.1 Training Dataset

- **Sources:** ICS logs of machine usage (load, feedrate, spindle speed)
- **Frequency of data:** Data clipped to 1-second intervals for consistency.
- **Filtering:** Only entries that meet the *machining-true* criteria are included.

4.2 Preprocessing

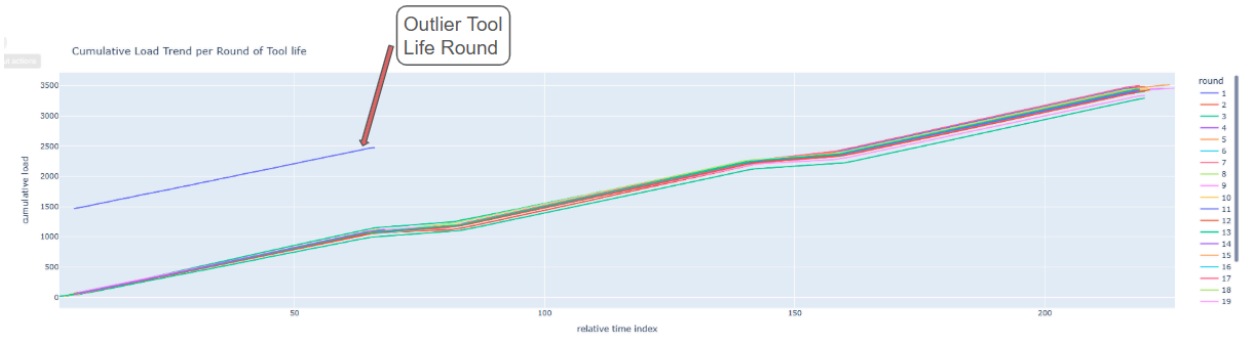
- **Feature Engineering:** Rolling window statistics are created to provide temporal context (capturing short-term trends in load, feedrate, etc.)
- **Normalization:** Cumulative load is scaled to 0–100% RUL, where each tool run is normalized based on its own maximum cumulative load.
- **Dynamic Outlier Detection:**

Outlier Optimization

A critical step before training is **Dynamic Outlier Detection**:

- Example problem: Operator changes a tool on the shop floor but forgets to log it on the application → unrealistic tool lives (like, 5000 load points instead of 2000).
- **Solution:** Dynamic Time Warping (DTW) used to detect abnormal wear curves.

- Remove any outlier runs to avoid using them in training the model



4.3 Training

- **Initial Training:**
  - If an **already existing Machine-Program-Tool id**: The model is first trained on historical tool lifecycle data, capturing patterns of wear across multiple machine-program-tool combinations.
  - If **a new combination comes in**: The model will take the first 20 rounds of tool changes (checked for validity dynamically) for training a new model.
- **Deployment Strategy:** A separate model is trained for each Machine-Program-Tool combination to take into account the behavioral differences for every tool across programs and even machines.
- **Evaluation:** RMSE is measured per unique tool run to check the model's adaptability.

4.4 Inference Pipeline

4.4.1 Overview

- **Prediction Frequency:** Every 1–10 seconds during machining.
- **Input Assembly:** Current machining state (load, feedrate, spindle speed) + recency features (rolling statistics).
- **Output:** RUL % — remaining cumulative load capacity.
- **Final Result to User:** The model's predictions will be translated to
  - **predicted ppe**,
  - a **reliability score,**

- **Tool Wear Classification**

4.4.2 Final Result to User

Within each tool run/life:

1. **Predicted PPE**
   - **Initial Setting:** Say, the operator-set PPE is 5 parts (i.e., tool is expected to make 5 parts in its full life).
   - **At a Point in Time (t):**
     i. Model Prediction: 63% RUL remaining.
     ii. Parts Completed so far: 3 parts (out of the set PPE = 5).
   - **Interpretation:**
     i. The tool has already used 37% of its life to make 3 parts.
     ii. With the remaining 63% life, the tool should be able to make 5 more parts.
   - **Updated Predicted PPE:**
     i. Already completed: 3 parts
     ii. Expected additional parts: 5 parts
     iii. Total predicted PPE = 3 + 5 = 8 parts

   Limitations:

   These calculations can be made only when the tool's parts-made value increases.

   1. We can't translate the model's predictions until the part is complete
   2. because at any point we don't know how much of a part is made, like whether 50% of the part is completed or not.
   3. So we can't show model predictions in terms of ppe in real-time.
2. **Reliability Score**

   A metric that measures consistency of predictions across the base models inside the ensemble (LightGBM, Random Forest, XGBoost).

   How It’s Calculated

   - For each validation fold:

i. Each base model predicts RUL% on the same validation samples.
ii. The standard deviation of these predictions is computed per sample.
iii. The mean of these deviations across all samples = the fold's reliability score.

- If all base models agree **(low variance in predictions) → high reliability.**
- If base models disagree strongly **(high variance) → low reliability.**
- For example, sample predictions for one tool state:
  i. LGBM = 60%, RF = 62%, XGB = 59% → Std ≈ 1.2 → High reliability.
  ii. LGBM = 40%, RF = 55%, XGB = 70% → Std ≈ 12.4 → Low reliability.
- The score is averaged across folds in cross-validation.
- **Output:** avg_reliability_score = a single value summarizing model prediction stability.
- This will provide a trust indicator beyond error metrics (RMSE, MAE).
- Helps flag situations where predictions are numerically correct but inconsistent across models and help warn the users of uncertainty.

3. Tool Wear Classification

- Will move ahead with the current implementation of Tool Wear Classification into Low and High Wear
- Which is based on the comparison of set_ppe by the operator and our predicted_ppe.
  i. If predicted_ppe > set_ppe, then Low Tool Wear.
  ii. If predicted_ppe < set_ppe, then High Tool Wear.

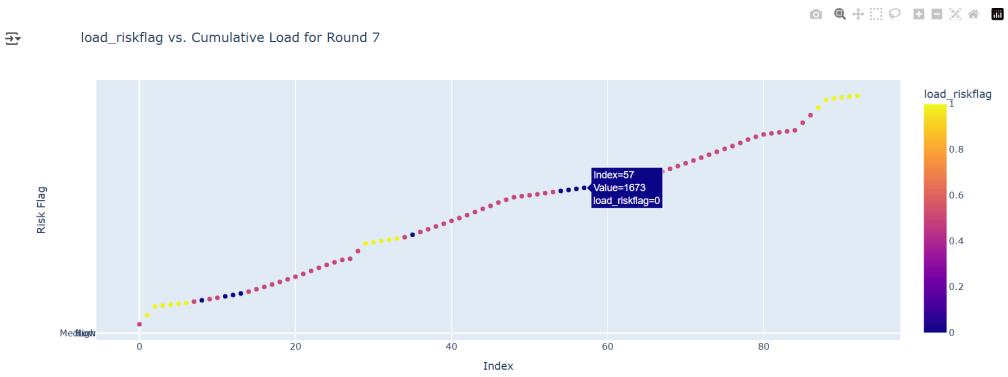
Attempt to implement this using an ML model:

1. Why?

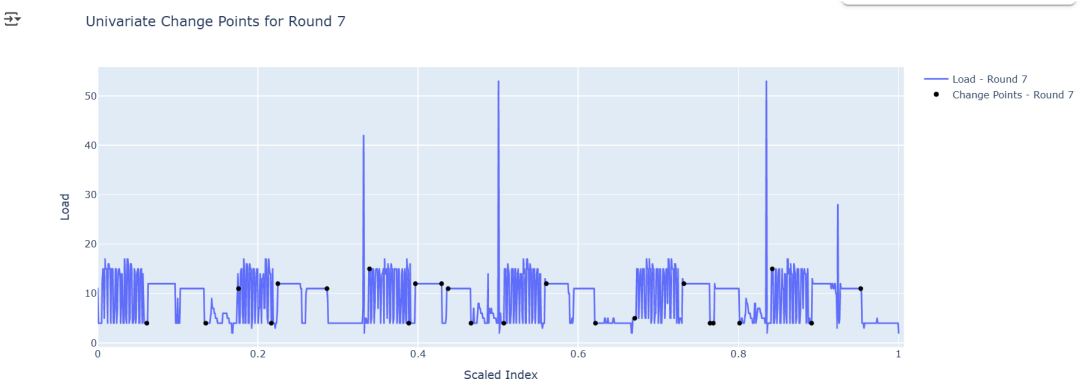
    a. The model will have more context of other machining parameters like load, feed, spindle speed and even the temporal trends embedded in the ft. eng. step.
    b. So that this classification is not just dependent on the predicted_ppe values.
2. Objective defined: To classify the tool life into three incremental stages of Low/Medium/High Wear.

Trials:

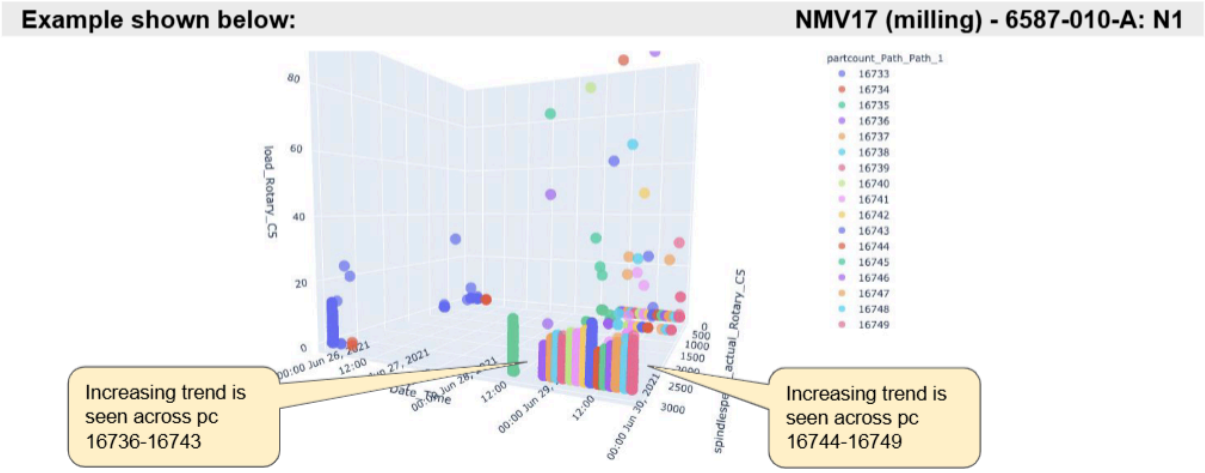
1. **Defining thresholds of change** for each tool based on:
    a. 20 rounds of data that will be used for model training
    b. Used the following principle to define aggregated thresholds for each machining variable.
        i. The avg and std in the data so far (or past 5 or 50 secs window)
        ii. **Approach 1:** IF the current_load > AVG(load in past window) AND the STD(in window due to current load) - STD(in prev window) > 0.25 * IQR (till now)
        iii. **Approach 2:** Using IQR, AVG, STD in the past window, define thresholds for low and high (in bet is med.)



2. Tried a **ruptures** library used for **change point detection** (another statistical technique)

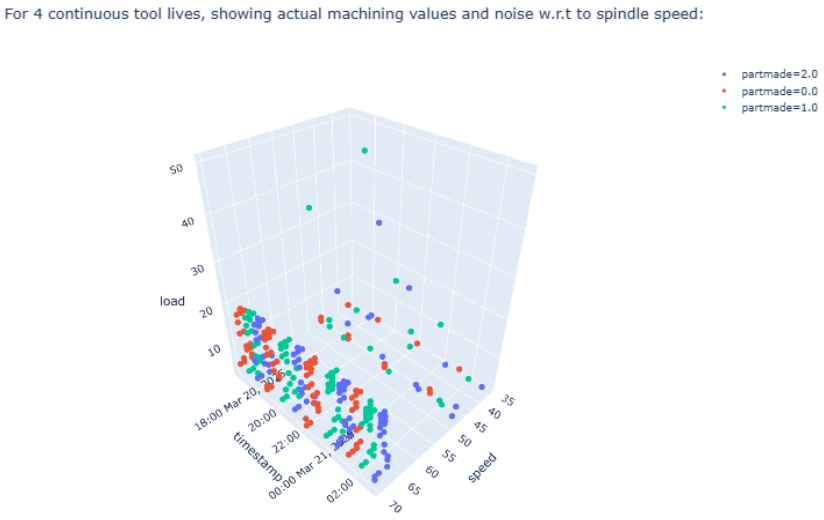


3. **Load Analysis from the Spindle Speed Perspective:**
    - From a previous experiment done on Tool Life on Leesta's data (referred ppt attached: LF_LEESTA_JIT...):
        - It was seen on pg 12 that most of the machining by a tool is done around a specific spindle speed.
        - When looking at the load values on a per-part basis at this spindlespeed, it was noted that there is an increase in the load values accumulated on tool over parts made. (referred graph attached below)

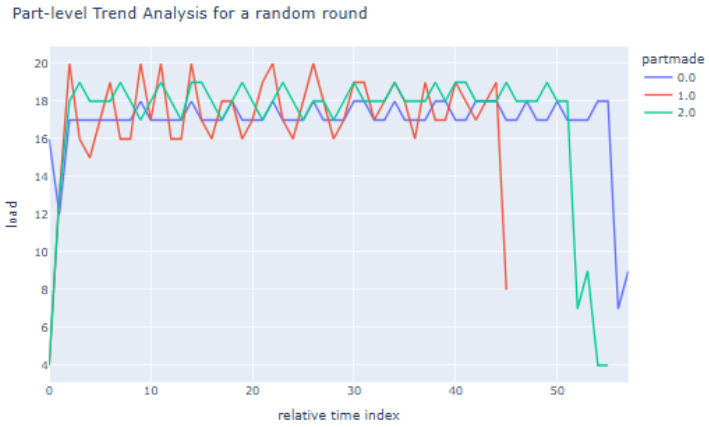


- So the objective is to be able to see this in the current dataset available. If we can find such a pattern here, then we will be able to build a modelling approach here.
- **Result**: Could not find an increasing trend in load values over parts made by a tool in a single tool life round.
- Example: A sample tool's data analysis here:

- The graph below shows a sample tool's 4 consecutive tool lifes.
- The series of red-green-blue represents one tool run/life.



- Checking the trend of increasing load values in our current datasets:



- No trend of increasing load points.

- This means that we will not be able to define robust thresholds to divide the tool life into Low/Medium/High.
- **Conclusion**: Tough to define thresholds for Tool Wear Classification task by a statistical/modelling approach currently. So we stick to our current implementation.

4.5 Feedback and Model Update

At the end of each tool run/life:

1. **Ground Truth Capture:** The actual tool change point (max cumulative load) is recorded.
2. **Validation:** Predictions are compared against the true value to assess accuracy and confidence.
3. **Retraining:** The full run (features + actual RUL) is appended to the dataset. The model is then retrained or incrementally updated, improving its understanding of tool wear under real conditions.
4. **Frequency of Retraining:** Can be done after every 10 tool lives or after every tool life. This frequency still needs to be decided upon when working on the deployment planning and real cost analysis of the entire end-to-end process.

So finally, all this cycle repeats for every new tool run:

- Predict RUL during the run
- Capture ground truth at tool change
- Update training dataset
- Retrain the model for better future predictions

5. Estimated Cost Analysis

The detailed cost analysis can be found in the following spreadsheet:

Tool Life Model Cost Evaluation

6. Limitations and Assumptions

- Predictions are **trend-based**, not absolute or based on any theoretical calculations – outputs should be used for guidance.
- The model is **machine-program-tool specific** (does not generalize across all).

- Real-world factors like coolant flow, vibrations, and thermal drift are **not modeled** here. (Leesta's dataset does not include those parameters currently.)
- Requires periodic retraining as tool behavior may drift with usage by the user.
- ML models are less transparent; interpretability requires SHAP/LIME analysis.

7. Future Improvement Scope

- **Reducing Training Data Requirement:** Currently, we need 20 rounds of tool changes to achieve acceptable error rates of 2-4%. But this may not be feasible for tools with larger ppe values. So try to reduce this requirement in further model enhancements.
- **Explainability Enhancements**: Incorporating SHAP within the pipeline for per-feature contribution tracking.
- **Stabilized Predictions**: Temporal smoothing or probabilistic filtering to avoid jitter.
- **Survival Analysis Integration**: Experimented with but not conclusive, may need further work as we acquire different machining params like temperature, vibrations, etc.