Journal of Intelligent Manufacturing
https://doi.org/10.1007/s10845-026-02867-7

Development of a semi-automated data acquisition and processing architecture for machine learning applications in grinding

Eike Reuter[1] · **Björn Killich**[1] · **Peter Breuer**[1] · **Sebastian Prinz**[1] · **Thomas Bergs**[1,2]

Received: 7 October 2025 / Accepted: 30 March 2026

Abstract
Studies show that manufacturing companies that invest in automation and artificial intelligence can achieve productivity gains of up to 20% in production output and labor efficiency, driven by advances in data analytics and machine-learning technologies. In grinding processes, sensor-based process variables such as accelerations and acoustic emission signals are commonly used as inputs for machine-learning models to predict quality-related outcomes, including surface roughness and structural damage. However, industrial adoption is often limited by the lack of data acquisition and storage architectures tailored to data-driven applications, as well as by the high computational demands of many modeling approaches in production environments. To address these challenges, this work presents a modular architecture for grinding process monitoring combined with an efficient feature extraction and selection methodology based on WELCH spectral analysis. The architecture enables structured data acquisition, centralized data storage, and consistent metadata management, while separating data processing, modeling and application layers. This modular design supports traceability and seamless integration of machine-learning models into industrial monitoring. Within this framework, physically interpretable frequency-domain features are extracted and systematically optimized. A hyperheuristic feature selection strategy is evaluated and compared with filter-based methods and a standalone Genetic Algorithm. The results show that filter-based approaches suffer from strong overfitting and limited generalization, while a standalone Genetic Algorithm improved robustness and predictive performance compared to filtering but remained dependent on problem complexity and population size. In contrast, the hyperheuristic consistently achieved superior robustness and predictive performance with reduced variance under blocked cross-validation.

Keywords Data acquisition · Data processing · Grinding · Machine learning

Introduction

Recent studies show that manufacturing companies that invest in automation and AI experience productivity increases of up to 20% in production output and labor efficiency. These improvements are made possible by data analytics, machine learning and other technologies (Deloitte, 2025). Machine learning models are also becoming increasingly important in grinding technology. One indicator of this is the growing number of grinding machines equipped with systems that collect data from machine controls and sensors, as well as the increasing number of publications in this field (Krajnik et al., 2024). In an industrial environment, machine learning models offer advantages in areas where physical and simple empirical models are weak. Unlike physical models, they require less fundamental process understanding. Advantages over finite element (FE) and molecular dynamics (MD) models include lower start-up and calculation effort (Brinksmeier et al., 2006). Given the increasing availability of data during process execution, machine learning models offer new possibilities for process monitoring. Process state variables, such as forces, accelerations, and acoustic emission signals are used as input for models that predict process result variables, such as workpiece roughness and structural damage (Pandian et al., 2020). However, a challenge in training and using these models in an industrial environment is

✉ Eike Reuter
e.reuter@mti.rwth-aachen.de

[1] Manufacturing Technology Institute – MTI, RWTH Aachen University, Campus-Boulevard 30, 52074 Aachen, Germany

[2] Fraunhofer Institute for Production Technology IPT, Steinbachstraße 17, 52074 Aachen, Germany

Published online: 10 April 2026

the lack of data acquisition and storage architectures developed specifically for interaction with data-driven models. Conversely, many models developed thus far cannot be used due to the use of computationally intensive algorithms in the direct production environment with partly local computing units (Khouas et al., 2024). Small and medium-sized enterprises (SMEs) are particularly affected by this. As a result, around 22% of SMEs are still not involved in machine learning and only around 10% actively use machine learning, even though 29% see machine learning as an opportunity for innovation (Burggräf et al., 2024). Building on deficits in data acquisition, processing, and modeling, this publication presents a holistic approach from data acquisition to data-driven modeling for live applications in grinding technology.

State of the art

The increased use of sensors in machine tools for process monitoring requires an efficient means of acquiring and storing data. In terms of manufacturing technology, it is particularly necessary to combine the acquisition and storage of metadata and time series data (Beckers et al., 2022). Metadata is generally defined as data that describes other data (Furner, 2020). In the field of manufacturing, metadata provides background information on recorded time series data. It specifies information about the component, the tool, or the manufacturing process. Time series data are states recorded at discrete points in time (Esling & Agon, 2012). In production technology, time series data are used to record machine control or sensor signals. Due to their different structures and the amount of data they generate, efficiently storing both types of data within a single storage system is not possible. Metadata usually accumulate in small quantities and is stored in structured databases. These databases offer high data integrity, sufficient performance, and good vertical scalability. Pre-processed time series data are primarily stored in data warehouses but lose important information through preprocessing. Because of their size, unprocessed time series can only be stored as raw data in data lakes (Mathis, 2017). However, due to the unstructured nature of these storage systems, time series data can lose information content when decoupled from its metadata. Additionally, due to the unstructured nature of data lakes, complex queries are necessary, which limits performance (Sawadogo & Darmont, 2021). Because of these issues, various time series storage solutions have been developed, but they are not directly transferable to the manufacturing industry (Jensen et al., 2017).

In the field of manufacturing technology, data collection is primarily related to the digitalization of process chains and digital twins. BECKERS ET AL. emphasized the importance of effective data management for sustainably using collected data in manufacturing. Therefore, they derived the need for a standardized database system for metadata and time series data according to the FAIR principles. However, detailed insights into data acquisition and generating data-driven models based on it are lacking (Beckers et al., 2022). GANSER ET AL. developed a digital twin framework for milling. Using a lambda architecture for data acquisition and microservice integration for modeling, they created a digital twin that maps profile deviation during the milling of blisks. While the research approach is promising, the modeling used is not explained in detail and cannot be applied directly to the grinding process (Ganser et al., 2021). Other approaches to creating digital twins in production technology mainly relate to production planning and control. However, due to the exclusion of metadata for process planning and time series data for process monitoring, these approaches are not sufficiently transferable to the current use case (Huang et al., 2022).

The number of research approaches developed in the field of data-driven modeling in grinding technology continues to increase. Currently, the formation of models is mainly used for predicting process results using supervised machine learning models. Current research focuses on creating classification models to detect grinding burn or grinding wheel wear (Krajnik et al., 2024). The process of generating supervised machine learning models usually involves five steps. First, process data is recorded in the form of internal drive data or external sensor data as a raw signal. In production technology, the focus is on acquiring time series data from spindle power, force, acceleration, and acoustic emission (AE) sensors. (Pandian et al., 2020).

The second step involves optionally preprocessing the raw signals by segmenting or filtering them. This increases the information content of the process data by eliminating idle times or interference frequencies caused by external influences or drives, for example.

The third step involves converting the continuous time signal of the process data into features from the time, frequency, or time–frequency domain during feature extraction. During time-domain feature extraction, features such as the root mean square (RMS), variance, skewness, and kurtosis of the raw signal are considered (Barandas et al., 2020). Feature extraction in the frequency domain is usually carried out using an upstream Fast Fourier Transform (FFT) of the raw signal. Time- and frequency-dependent feature extraction is achieved by transforming the raw signal into the time–frequency domain. Examples of this include the Short-Time Fourier Transform (STFT) and discrete wavelet decomposition (DWD). (Mahata et al., 2021). In addition to classical, analytically derived transformation methods, heuristic and bio-inspired principles such as swarm decomposition are also applied in the damage diagnosis of machines and components (Vashishtha et al., 2021).

The fourth and fifth steps are feature selection and model training/validation. In feature selection, features with high correlation to the target variable and low correlation to other features are selected as model input. Supervised machine learning models are formed in training and validation phases. A machine learning model receives an input dataset and the corresponding result variables. During the training phase, the correlations and dependencies between the input and result data are determined and the mathematical model is formed. After training, the algorithm is applied to unknown data for validation. (Rebala et al., 2019).

Regarding the data-driven modeling of grinding wheel wear, hidden MARKOV models (Sachin Krishnan & Rameshkumar, 2021) and decision tree models (Mouli & Rameshkumar, 2020) were trained using features from the time domain. (Warren Liao, 2010) and (Yang & Yu, 2011) transformed the AE signal into the time–frequency domain and used the wavelet coefficients as features. Autoregression models (ARs) and support vector machine models (SVMs) were used for modeling. Most approaches in the field of grinding burn detection were developed to identify individual grinding burn states. Methods for extracting features from the time domain, the frequency domain (Yang & Yu, 2013) and the time–frequency domain (Sauter et al., 2021) were chosen. A detailed overview of other approaches can be found in (Pandian et al., 2020) and (Krajnik et al., 2024). In addition to the achieved model scores, scientific work to date shows that, in most cases, AE-signals are superior to spindle power, forces, and accelerations in terms of their information content. Furthermore, combining several sensors and preprocessing, such as filtering and feature selection, increases model scores. Models such as SVMs and further linear regression developments are superior to artificial neural networks (ANNs), especially for smaller data sets.

A central limitation of previous research on data-driven modeling lies in its restricted suitability for economic and industrial deployment. While existing approaches often include fundamental methods for data acquisition and model development, their transferability to other grinding processes, scalability to larger data volumes and applicability in live industrial environments remain limited. This is partly due to the absence of data acquisition and storage architectures specifically designed to interact with data-driven models and partly because many proposed models rely on computationally intensive algorithms that are impractical for use on local production hardware. Moreover, existing architectures typically lack a systematic framework for enabling future model retraining, transfer learning or domain adaptation, which are essential for maintaining model performance under changing process conditions.

Based on the aforementioned shortcomings in data acquisition, processing and modeling, this publication presents a comprehensive and scalable approach to data-driven modeling for live application use in grinding technology. Beyond addressing current limitations, the proposed architecture is designed to systematically support industrial deployment, scalability to larger data volumes and transferability across different grinding processes. The status quo and limitations of each building block within a newly developed end-to-end architecture are discussed step by step and novel concepts for data acquisition, feature extraction, model optimization and validation are introduced. In particular, the approach explicitly enables robust model validation, future retraining and domain adaptation by integrating structured metadata management and modular processing layers. Finally, the methodology is validated using a grinding burn prediction use case in surface grinding, demonstrating both its practical applicability and its potential as a foundation for long-term, adaptive machine-learning solutions in industrial environments.

Development of the data acquisition and processing architecture

Based on the state of the art, it was identified that the efficient and parallel acquisition of metadata and time series data is insufficiently addressed in existing research. Beyond this core requirement, four additional design criteria were defined for the proposed approach. The overall architecture incorporates a structured metadata database while deliberately following a decentralized design paradigm. Moreover, it enables the parallel integration of machine control data and external sensor signals and is inherently scalable, allowing straightforward extension to additional machines and data sources.

Building blocks of the architecture

The architecture developed for data acquisition and processing is composed of eight interacting components that are connected via distributed systems and standardized communication interfaces, as illustrated in Fig. 1. At the machine level, each grinding machine is linked to both a metadata database and a data lake through a dedicated data acquisition system. This system acquires machine control data as time series directly from the machine tool using industrial communication protocols such as OPC UA or PROFIBUS and stores the data in the data lake. In addition, machine tools may be equipped with external sensors, including acoustic emission or acceleration sensors, whose analog signals are amplified, digitized via measurement cards and stored as time series alongside the machine control data. The data lake serves as a centralized, largely unstructured repository for all recorded time series in their raw format. In contrast, the metadata

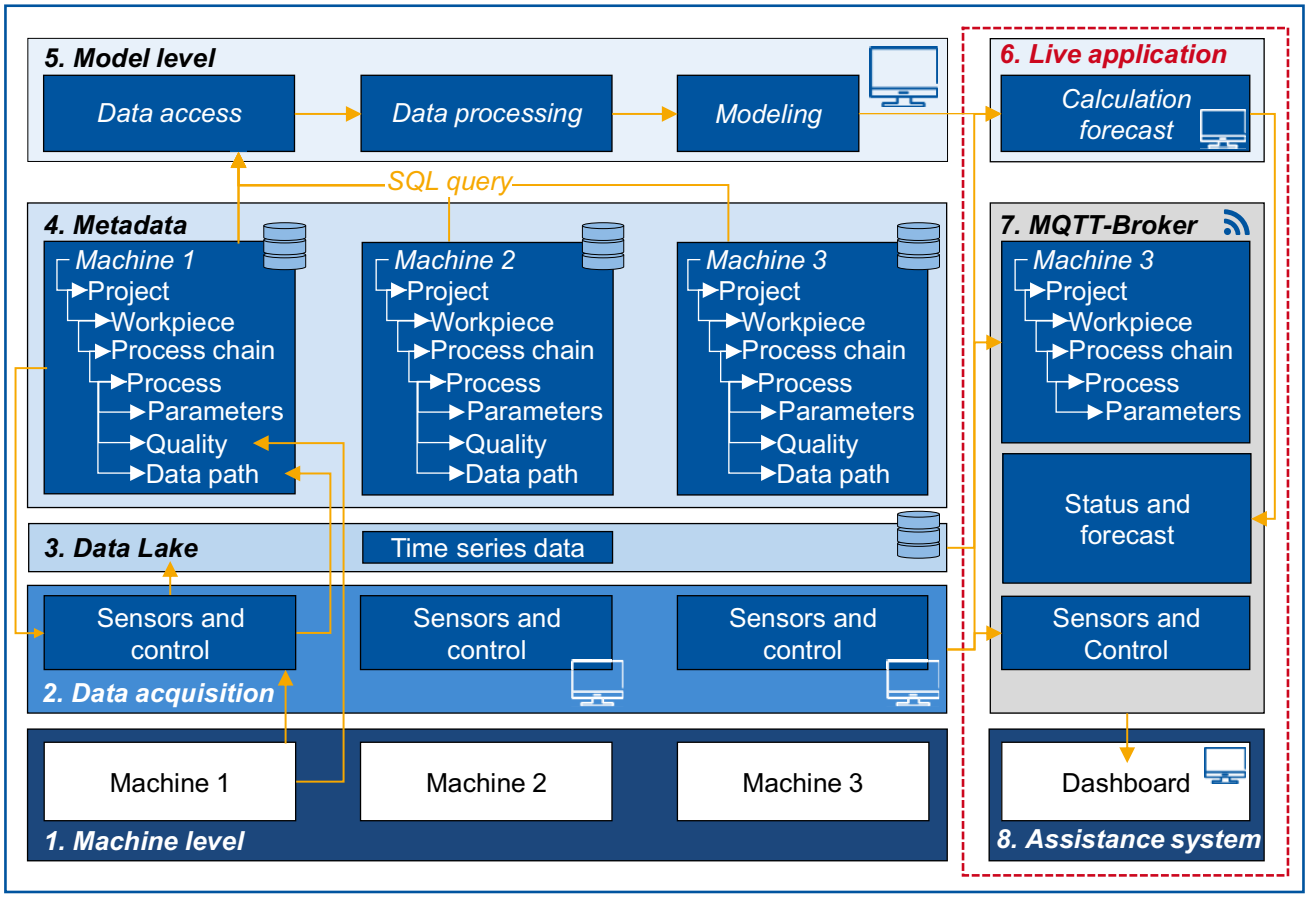

Fig. 1 General data acquisition and processing architecture

database is implemented as a structured relational database that describes production processes and components through project identifiers, materials, workpiece information and process parameters. For each completed grinding process, the data acquisition system generates a unique storage path for the corresponding time series and stores this reference in the metadata database, thereby ensuring unambiguous linkage between metadata and raw data. At the model level, all recorded data can be accessed via a user interface by querying the metadata database, providing simultaneous access to both process descriptors and associated time series. These data form the basis for training machine-learning models, which are subsequently deployed in live process monitoring. A dedicated computing unit performs live predictions and communicates the results to a visualization dashboard via an MQTT broker, enabling continuous monitoring of the grinding process.

To enable stepwise training of machine-learning models for process monitoring, grinding process data must first be acquired in a structured and traceable manner. Within the proposed architecture, grinding operations are defined in a technology-specific way and represented digitally in the metadata database as modular building blocks of a grinding process chain. Individual links of this chain may comprise, for example, consecutive roughing and finishing operations, each characterized by their respective process parameters. Following process planning, all metadata required for manufacturing a component batch (e.g. process steps, machine tools, tooling information, and setup parameters) are stored persistently in the metadata database. At the beginning of production, the machine operator retrieves the relevant process definition for the component via a machine-side interface to the metadata database, thereby initializing the execution of the planned process chain. During machining, the data acquisition system records control and sensor signals as time series using predefined sampling strategies. After completion of the manufacturing process, the recorded time series are linked with the corresponding process metadata and consolidated into a single data object. This object is stored in the data lake under a uniquely generated identifier, which defines an unambiguous storage path for the resulting file. The same storage path is then written back to the metadata database, ensuring persistent referential integrity between metadata and raw time series data. For subsequent analysis and model development, data queries are performed via the metadata database by selecting relevant manufacturing processes, workpieces,

processes and their associated storage paths. The corresponding machine control and sensor data are then retrieved by resolving these paths in the data lake, enabling reproducible and scalable access to consistently contextualized process data.

The developed architecture implements a two-stage workflow for the training and deployment of machine-learning models. In the first stage, the acquired process data are preprocessed through systematic segmentation and feature extraction, as described in Chapters 4.1 and 4.2. In the second stage, supervised machine-learning models are trained and validated within an iterative feature selection and optimization procedure (see Chapters 4.3 and 4.4). These models predict grinding process result variables such as surface roughness, tool wear, and edge zone conditions based on sensor signals and machine control data. After successful training and validation, the models are stored and made available for use in live process monitoring applications.

During live operation, process state monitoring is realized via a dashboard that visualizes selected time series data, associated metadata and result variables predicted by pre-trained machine-learning models. To enable efficient visualization, the data acquisition system downsamples the recorded time series to an appropriate temporal resolution and transmits them to the dashboard via an MQTT broker. In parallel, the data acquisition system publishes a notification to the MQTT broker upon completion of a grinding process, indicating the availability of new data for inference and providing the corresponding storage path in the data lake. A dedicated computing unit for live calculation continuously subscribes to the MQTT broker to identify newly announced storage paths. Using these references, the computing unit retrieves the required time series data directly from the data lake, thereby avoiding the transmission of raw data through the MQTT messaging infrastructure. Live inference is subsequently performed using the pre-trained machine-learning models. The resulting predictions are published back to the MQTT broker and subscribed to by the dashboard, where they are visualized to enable continuous and systematic assessment of the grinding process status.

The developed architecture directly addresses several challenges related to industrial deployment, monitoring, and long-term robustness of machine-learning models. By structuring the data flow from the machine level through data acquisition, a centralized data lake and a harmonized metadata layer, the architecture ensures traceability, reproducibility, and consistent contextualization of sensor data across machines, projects, and process chains. This structured separation enables reliable model training and evaluation while preserving the ability to analyze condition-specific effects such as different tools, workpieces or parameter settings.

At the model level, the modular design of data access, processing, and modeling facilitates systematic retraining and controlled updates of ML-models as new data become available. The integration of a live application layer with live calculation and forecasting, coupled with an MQTT broker, allows continuous monitoring of model outputs and process states in production. This setup provides a natural entry point for detecting performance degradation and data drift by comparing live predictions with historical distributions and quality outcomes stored in the metadata layer.

The presented architecture should primarily be understood as a conceptual and methodological contribution rather than a finalized system optimized for a specific performance target. The focus lies on demonstrating how industrial process data can be systematically acquired, stored and analyzed in a post-process context using a coherent and maintainable architecture. By abstracting from application-specific optimizations, the approach highlights general design patterns and architectural decisions that can be transferred to a wide range of industrial use cases. In this sense, the contribution provides a structured framework for reasoning about data-driven process monitoring architectures rather than a prescriptive or benchmark-oriented solution.

Detailing the data collection

The state of the art shows that sensor fusion, i.e., the combination of multiple signal sources, in particular offers advantages in many applications in terms of increased model quality. For this reason, the architecture presented allows both machine control signals and acoustic emission, force or acceleration signals to be recorded synchronously. The control and sensor data are transmitted as a time-continous signal of the form:

$$\boldsymbol{xt},\ \boldsymbol{x} \in \mathbb{R} \tag{1}$$

With a sampling rate f_s and a resulting sampling interval of

$$\boldsymbol{\Delta t} = \frac{1}{\boldsymbol{f_s}} \tag{2}$$

the continuous time signal is divided into discrete, equidistantly distributed points in time

$$\boldsymbol{t_n} = \boldsymbol{n} \cdot \boldsymbol{\Delta t},\ \boldsymbol{n} \in \mathbb{N}_0 \tag{3}$$

which results in a time series of the original signal of the form:

$$\boldsymbol{x}[\boldsymbol{n}] = \boldsymbol{x}(\boldsymbol{n} \cdot \boldsymbol{\Delta t}),\ \boldsymbol{n} = 0,\ 1,\ 2,\ \ldots \boldsymbol{N} - 1 \tag{4}$$

To achieve this, the connected machine tools were equipped with TYROLIT TOOLSCOPE data acquisition systems to record the signals from the machine controls. The data are read via PROFIBUS interfaces with a sampling rate of $f_{s,C}$

= 100 Hz. In addition, the machine tools were equipped with CompactDAQ hardware from National Instruments to sample all sensor signals. NI CompactDAQ modules with maximum sampling rates of up to $f_{s,AE} = 1$ MHz were used to integrate acoustic emission sensors. NI CompactDAQ modules with sampling rates of $f_{s,F} = 2$ kHz and $f_{s,A} = 40$ kHz were used to record forces and accelerations.

Semi-automated data acquisition takes place via the data acquisition system, which has access to the metadata database and data lake and communicates with a live dashboard via an MQTT broker (see Fig. 2). The data acquisition system is based on a NI LabVIEW environment developed in-house. Acquisition of data for a grinding process begins with creating a base file containing the metadata of the grinding process. Once the file is created, the data acquisition system waits for a trigger signal to record the time series from the machine control and external sensors. The trigger signal is activated by an M command in the NC code of the grinding machine. When the trigger is activated, recording of time series data from the machine control and sensors begins. For performance reasons, data packets from all channels are written to the base file step by step. After each data packet is written, the system checks if the trigger signal is still active, which continues the time series recording. In addition to saving the time series, downsampled control and sensor data is simultaneously sent to the MQTT broker in JSON format. The dashboard can retrieve this data and display it live or it can be used for applications that require less computing power. As soon as another trigger in the NC code stops the recording of the time series, the entire data set is saved in the data lake and the storage path is stored in the metadata database and send to the MQTT broker. This architecture is particularly suitable for monitoring processes in frozen industrial environments, in which processes are qualified and no changes are permitted, so closed-loop control is not possible (e.g. aerospace industry) (ISO 10007, 2017). For example, it enables the immediate identification of grinding wheel wear or trends in workpiece roughness development after the process ends. Based on this information, one can make decisions regarding adaptive dressing cycles or identify workpieces out of tolerance. Unlike the aforementioned lambda architectures, the developed method is not real-time capable but easier to implement and maintain for industrial processes.

Development of the data processing and modeling architecture

The data processing and modeling architecture is based on three fundamental building blocks (see Fig. 3). First, all meta and time series data necessary for subsequent modeling is accessed. This data is accessed via a user interface with the relational metadata database described in Chapter 2. The database is then used to retrieve the necessary time series data from the data lake for modeling. After the data is provided, machine learning models are trained and validated in the data processing and modeling process. These models can then be used in a live application on the machine.

The models are trained following the general procedural structure established in the state of the art, comprising preprocessing, feature extraction, feature selection and subsequent model training. In the feature extraction stage, discrete and characteristic features are derived from the recorded time series and after appropriate selection, provided as inputs to the machine-learning models. While the stepwise training workflow commonly reported in previous studies (see Chapter 2) is retained, the underlying methods are substantially modified. The primary objective is to derive an efficient model structure that can be deployed in industrial environments with inference times on the order of seconds. To this end, five fundamental requirements are imposed on the processing architecture. Feature extraction methods are designed to enable high-resolution analysis in the frequency domain across both low and high frequency ranges. At the same time, the extracted features must be computationally efficient and physically interpretable, allowing direct insights into potential optimization of the sampling rate in Hertz. Furthermore, the feature selection strategy is explicitly tailored to reduce computational effort compared to conventional approaches, while preserving a high level of predictive performance and model quality.

Segmentation method

Time series data must be segmented to either reduce the amount of recorded data or increase its information content (Krajnik et al., 2024). One main reason for segmentation is to eliminate air grinding times, which are not correlated with the target variables in subsequent modeling. Other reasons include dividing a continuous signal into different, component-specific segments (e.g. the grooves of an end mill) and subdividing the process of grinding a component feature into individual steps (e.g. roughing and finishing). The presented architecture makes semi-automated segmentation of time series possible on the basis of various machine control data. The main prerequisite is time-synchronized recording of all machine control and sensor signals, converted via the sampling rate f_s. In this architecture, all time series data can be segmented based on spindle power, axis positions and axis speeds. In Fig. 4 the acoustic emission signal (AE-signal) of a pendulum grinding process is used to demonstrate segmentation based on the aforementioned control data. The example shown aims to subdivide the overall signal into individual strokes to allow for the separate

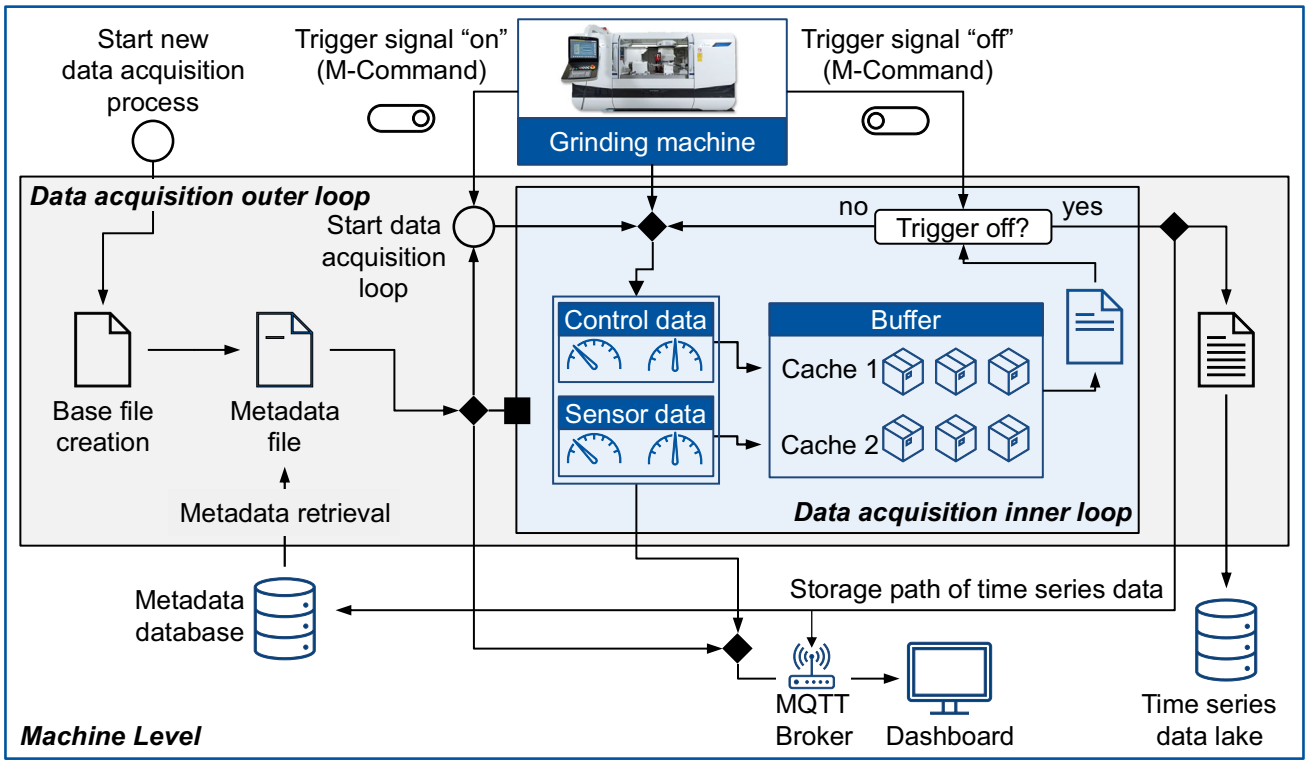

Fig. 2 Automated time series acquisition

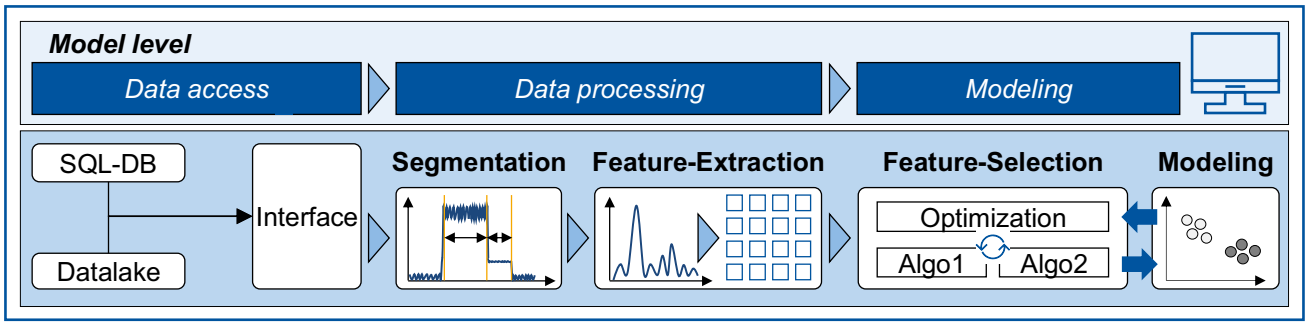

Fig. 3 General modeling procedure

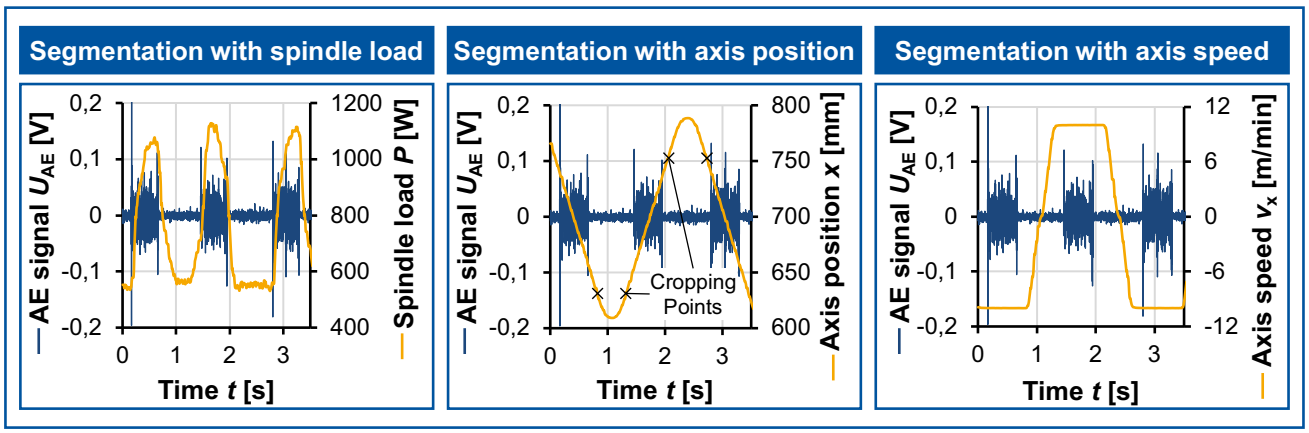

Fig. 4 Automated time series segmentation

consideration of the grinding segments in upgrinding and downgrinding.

The figure shows the AE signal U_{AE} with three grinding strokes over time. The different machine control data are plotted on the vertical secondary axis. To define a new segment, cropping points θ are defined within the machine control data by means of threshold value detection. Exceeding or falling below these values indicates the start or end of a segment. Based on a time series $x[n]$, the number of threshold values is calculated with

$$S_{over} = \left\{ n_i \in N | x(n_{i-1}) \leq \theta \bigwedge x(n_i) > \theta \right\} \tag{5}$$

in the case of exceedance and the number of underruns S_{under} by:

$$S_{under} = \left\{ n_i \in N | x(n_{i-1}) \geq \theta \bigwedge x(n_i) < \theta \right\} \tag{6}$$

Finally, the cropping points found can be transferred to all other time series with the aid of the respective sampling rate of the time series. For the subsequent feature extraction, the features are calculated for each of the segments.

Selection of the feature extraction method

Current approaches to train data-driven models in grinding technology primarily use feature-based methods. Here, the recorded time series is not used as direct input for the models, but rather characteristic features of these. The advantages of this method are high efficiency in modeling, good interpretability and the reduced risk of overfitting (Storcheus et al., 2015). Within a so-called feature extraction process, the time series $x[n]$ is transformed into a feature vector f_{V} (Fulcher, 2017):

$$x[n] \overset{Transformation}{\rightarrow} Feature - Vector f_V,\ f_V \in \mathbb{R}^k \tag{7}$$

The transformation process can include various preprocessing steps, such as the segmentation of the time series into time-divided intervals as described above, as well as the application of physical filters and conversions in the frequency and time–frequency domain. If the transformation process is summarized within a feature operator or a mapping function $\Phi(\mathrm{x})$, the resulting feature vector is called

$$f_V = \phi(x) = \begin{bmatrix} f_{V1}(x) \\ f_{V2}(x) \\ \vdots \\ f_{Vk}(x) \end{bmatrix} \tag{8}$$

where k is the number of extracted features (Christ, 2018). The current state of the art includes various approaches to data-driven modeling for preprocessing time series and calculating features. For this purpose, standard python libraries are generally used to calculate one feature type (e. g. pysignal) or several types (tsfel or tsfresh) (Barandas et al., 2020). One basic distinction is between extraction methods in the time, frequency or time–frequency domains. As discussed in Chapter 2, the initial scientific approaches to train data-driven models focused on calculating characteristic features in the time domain. Advantages of this approach include good interpretability of the resulting features and low computational and memory requirements. However, this approach requires knowledge-based preprocessing of the raw signal, e.g., filtering to compensate for possible interference signals. Additionally, the information content is low due to the lack of spectral evaluation of the raw signal (Krajnik et al., 2024). Due to these limitations, approaches that transformed the time-resolved raw signal into the frequency domain were increasingly pursued. The most common method for this is the discrete FOURIER transform (DFT). Here, each i-th frequency index is assigned a complex FOURIER coefficient FFT by the formula

$$FFT_x[i] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}in},\ i = 0,\ 1,\ \ldots N-1 \tag{9}$$

where e corresponds to the EULER number, j to the imaginary unit and N to the length of the time series (Vetterli & Kovačević, 2014). Based on the calculation of the individual FOURIER coefficients, the energy of the frequency bins, for example, can be calculated and used as input data for data-driven models. In practice, the evaluable maximum frequency $f_{\mathrm{max,FFT}}$ is limited by the NYQUIST-SHANNON theorem to (Landau, 1967):

$$f_{max,FFT} = \frac{f_s}{2} \tag{10}$$

The frequency resolution Δf_{FFT} is set via

$$\Delta f_{FFT} = \frac{f_s}{N} \tag{11}$$

and is distributed linearly over the entire frequency band (Oppenheim et al., 1999). Due to the lack of time resolution of the FOURIER transform, it is possible to extend the formula by shifting a window w over the signal in time in order to calculate time-resolved frequency information. In most cases, a Short-Time FOURIER Transform (STFT) is used for this purpose, so that

$$STFT_x[m,\ i] = \sum_{n=-\infty}^{\infty} x[n] \cdot w_{STFT}[n-m] \cdot e^{-j\frac{2\pi}{M}in},\quad i = 0,\ 1,\ \ldots M-1 \tag{12}$$

whereby the window w_{STFT} depends on the window shift m (Allen & Rabiner, 1977). The frequency index i indicates the frequency at which the transformation is performed and M represents the window length. Due to the constant window length M, derived from HEISENBERG's uncertainty principle, it is not possible to have a simultaneously high frequency and time resolution (Vetterli & Kovačević, 1995). Large values of M result in high frequency resolution and reduced time resolution. Reducing the window length increases the time resolution while decreasing the frequency resolution. Based on this limitation, a method of converting the raw signal into the time–frequency domain with variable time and frequency intervals was developed using wavelet transforms. By shifting and scaling a basic wavelet function ψ, wavelet coefficients W_{x} are defined by the mathematical relationship

$$W_x[s,\, b] = \int_{-\infty}^{\infty} x[n] \cdot \frac{1}{\sqrt{|s|}} \psi * \left(\frac{t-b}{s}\right) dt \tag{13}$$

where s represents the scaling parameter for frequency control and b the parameter for time shifting (Vetterli & Kovačević, 1995). The advantage of variable interval lengths in the time and frequency domains is often helpful. However, it is accompanied by the disadvantage that selecting suitable wavelet basis functions and estimating correct scaling resolutions is complex and, in many cases, requires knowledge of the empirical process. Converting the scales s into frequencies f in Hertz is possible via equivalent FOURIER periods, depending on the wavelet basis functions. A widely used wavelet basis function is the MORLET wavelet with

$$\psi(t) = \pi^{\frac{1}{4}} \cdot e^{-j w_0 t} \cdot e^{-\frac{t^2}{2}} \tag{14}$$

where ω_0 corresponds to the central frequency of the sine wave (typical for MORLET: $\omega_0 = 6$) and, derived from this, $e^{(-\mathrm{j})\,(\omega 0)\,(\mathrm{t})}$ corresponds to the sine wave itself in complex notation. The time-dependent component of the function is represented by the term $e^{(-\mathrm{t}2/2)}$. By converting the equivalent Fourier wavelength λ with

$$\lambda = \frac{4\pi s}{w_0 + \sqrt{2 + w_0^2}} \tag{15}$$

the scales s are converted into frequencies f using the relationship (Torrence & Compo, 1998):

$$f = \frac{1}{\lambda s} \tag{16}$$

Due to the aforementioned dyadic scaling of s to obtain variable intervals in both the time and frequency domain, the converted frequency resolution Δf_{W} is normally logarithmic. As a result, achieving a high spectral resolution at high frequencies is usually not possible or it requires a high computational effort and detours.

The spectral density estimation according to WELCH offers a further possibility of transformation into the frequency domain that has not yet been established in grinding technology. This method aims to reduce the variance of the spectral density of conventional FFTs while reducing computing and memory requirements (Welch, 1967). To obtain the spectral density, the time series $x[n]$ is first divided into K segments of length L, whereby the individual segments are divided by:

$$x_x[n] = x[n + k \cdot L],\, n = 0{,}1,\, \ldots,\, L-1 and k = 0,\, \ldots,\, K-1 \tag{17}$$

To prevent edge artifacts, a window function w (e.g. HAMMING window) is then applied so that:

$$x_k^w[n] = x_k[n] \cdot w_H[n] \tag{18}$$

In the next step, an FFT of the form

$$X_k[m] = \sum_{n=0}^{L-1} x_k^w[n] \cdot e^{-j\frac{2\pi}{L}mn},\, l = 0,\, 1,\, \ldots,\, L-1 \tag{19}$$

is applied, whereby for each of the segments a periodogram of the form

$$P_k[m] = \sum_{n=0}^{L-1} \frac{1}{UL} \cdot |X_k[l]|^2 \tag{20}$$

is created. Due to the window function $w[n]$, the spectral power is greatly reduced without the inclusion of the so-called energy normalization factor U and therefore does not correspond to the actual spectral power. Scaling with

$$U = \frac{1}{L} \sum_{n=0}^{L-1} w[n]^2 \tag{21}$$

counterarcts. Finally, the periodograms of all segments are averaged so that the spectral density estimate according to WELCH can be calculated using

$$P_{Welch} = \frac{1}{K} \sum_{k=0}^{K-1} P_k[m] \tag{22}$$

whereby a spectral power is assigned to each l-th frequency interval. Figure 5 illustrates the spectral density estimation procedure according to WELCH, employing an exemplary raw AE-signal from a grinding process. The time-resolved raw

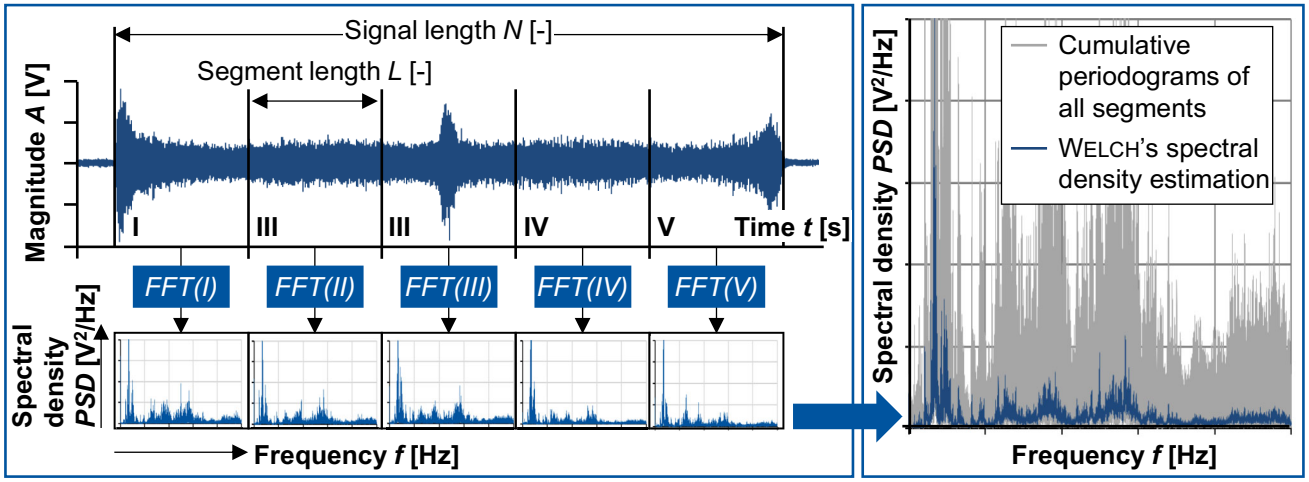

Fig. 5 Spectral density estimation according to WELCH

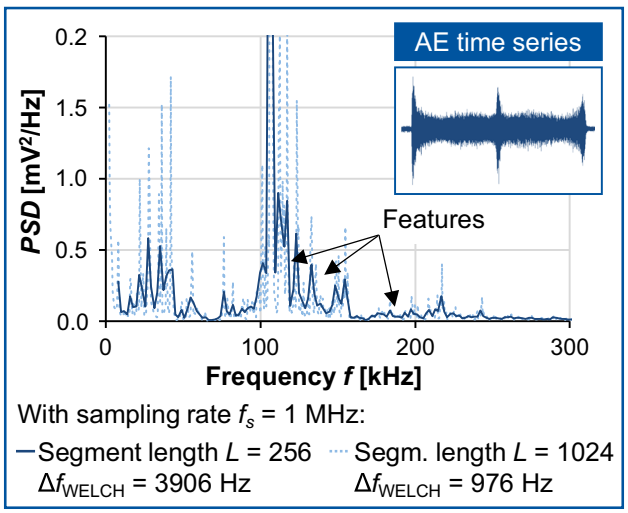

Fig. 6 Frequency resolution

signal with Magnitude A depicts a grinding stroke with a signal length of N. The total length of the grinding stroke is divided into five segments (I–V), each with a length of L. An FFT is applied to each of the individual segments, generating five periodograms. Averaging these periodograms produces the WELCH spectral density estimate shown on the right. For clarity, the window function is omitted. In practice, the individual segments are also overlapped to reduce the variance of the spectral density estimate further.

Formula 20 shows that spectral resolution depends on segment length L. As with the STFT, spectral resolution increases with longer segments, though the smoothing effect decreases. Figure 6 illustrates the relationship between segment length L and spectral resolution Δf_{Welch}. It shows the spectral density estimate PSD for a raw AE-signal during a grinding stroke. Due to the NYQUIST theorem, a sampling rate of $f_s = 1$ MHz enables a maximum frequency of $f_{\text{max,Welch}} = 500$ kHz, but the display is limited to $f = 300$ kHz for clarity. The spectral density estimate is clearly smoothed with a segment length of $L = 256$ compared to $L = 1024$. With the relationship

$$\Delta f_{Welch} = \frac{f_s}{L} \tag{23}$$

the frequency resolution Δf_{Welch} is calculated in the same way as the resolution of the STFT. The resolution is distributed linearly across all frequency ranges. For a segment length of $L = 256$, the resolution is $\Delta f_{\text{Welch}} = 3906$ Hz and for a segment length of $L = 1024$, the resolution is $\Delta f_{\text{Welch}} = 976$ Hz over the entire frequency spectrum. As with a FFT, the time resolution of the signal is generally omitted. Compared to a FFT, spectral resolution is reduced by spectral power density estimation according to WELCH. However, smoothing increases robustness against interference signals.

Regarding the requirements for the data processing and modeling architecture described in Chapter 1, the primary goal is to efficiently design the feature extraction process in an industrial environment using the necessary process-parallel application options. First and foremost, the computing and storage requirements must be minimized. The number of computing operations O is usually determined to estimate the computing requirements of the individual feature extraction methods. The computational effort of the methods presented here depends on the transformation algorithm and is shown as a function of signal length N and method-specific parameters. Fourier coefficients are typically obtained via a FFT, while wavelet coefficients are calculated using continuous wavelet transforms (CWTs), discrete wavelet transforms (DWTs) or wavelet packet transforms (WPTs). Figure 7 compares the number of arithmetic operations O as a function of various transformation algorithms and signal lengths N. The axes are plotted logarithmically. The blue graphs in the figure

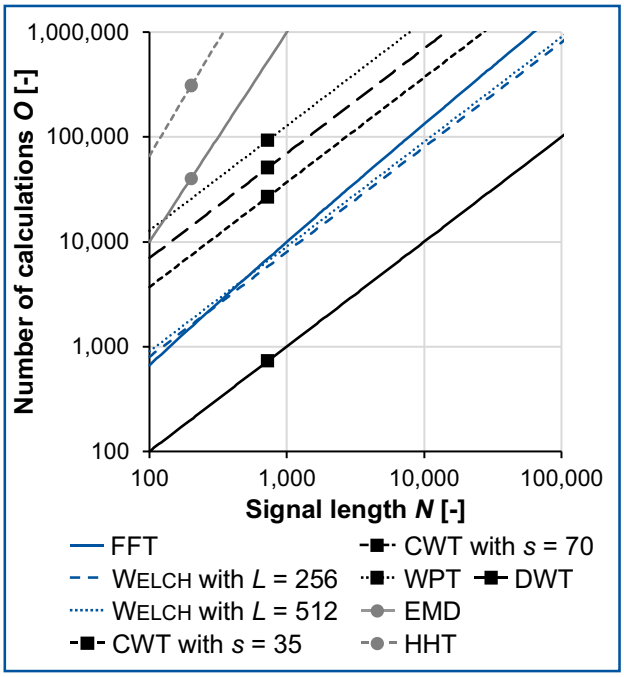

Fig. 7 Complexity of transformation algorithms

show the transformation methods based on simple FFTs. The black grouping of graphs corresponds to methods for calculating wavelets. Due to their mention in the state of the art, the Empirical Mode Decomposition (EMD) and Hilpert-Huang Transform (HHT) methods are also shown in gray. The complexity of calculating the FFT depends solely on the signal length N. In the worst case, this also applies to EMD and HHT (Elgamel, 2012). The computational effort of calculating STFTs and Welch's methods increases with the window lengths M and L or the number of scales s in relation to the wavelet transform. However, the computational effort for an STFT or Welch's method increases logarithmically with an increase in window size, whereas the computational effort for a continuous wavelet transform increases linearly with an increase in the number of scales (Wang & He, 2023). Both the window and segment lengths, as well as the number of scales, were chosen based on literature values (Torrence & Compo, 1998). Switching to the faster DWT greatly reduces the spectral resolution. Switching from a logarithmic to a linear resolution requires a WPT, which increases computational effort (Lei & Kun, 2017). EMD and HHT entail the highest computational effort because they increase linearly with signal length. Assuming the use of AE sensors with minimum sampling rates of $f_{s,\min} = 400$ kHz and manufacturing times of more than one second, transforming the raw signal in a few seconds is only possible with spectral density estimation according to Welch.

Based on previous findings, Table 1 summarizes the most important properties for selecting a feature extraction method and the degree to which the presented methods fulfill these properties. The plus signs indicate a positive fulfillment of the displayed characteristics. Minus signs indicate a negative fulfillment.

Due to the high computational effort involved, neither HHT, EMD, nor CWT can be used as feature extraction method in live applications. DWT requires the least computational effort. However, due to its lack of spectral resolution, the expected model quality is low. Transforming into the frequency domain using spectral density estimation according to Welch offers the lowest computational effort while providing high frequency resolution, which is distributed linearly over the entire frequency spectrum. The smoothing effect also enhances the robustness of the models. Since frequencies are represented directly in Hertz, the models are highly interpretable. This enables the estimation of important frequency ranges in model analyses and the reduction of sampling rates. This is particularly beneficial for small and medium-sized enterprises with limited storage capacities.

Based on the aforementioned advantages of spectral density estimation according to Welch, it was selected as the basis for the feature extraction method. In addition to the previous explanations, a Hamming window with a window overlap of 50% was selected for the spectral density estimation. In the method used, the total number of features n_f depends on the window length L of the spectral density estimation and the number of previously determined segments $n_{segments}$. Including the sampling rate f_s and the Nyquist theorem, the method presented therefore produces a total number of features of

$$n_f = \frac{L_{welch}}{2} \cdot n_{segments} \tag{24}$$

and is made available to the subsequent machine learning model as input. The input is provided as a feature matrix F, where the number of rows stands for the number of test points m_t and the spectral densities are plotted column by column. Figure 8 shows two examples of feature matrices as a function of the influencing factors presented.

In university research and industry, particularly in toolmaking and small-batch production, the number of features often exceeds or equals the number of test points. This increases the risk of overfitting, in which the model learns and reproduces random patterns, even when cross-validation is used. To minimize this effect, the literature recommends limiting the feature-to-test point ratio to 5:1 for simple models (Theodoridis, 2009). Because spectral density estimation results in a high number of features, the feature matrix must undergo feature selection to reduce the number of features. The objective is to minimize the number of features while maximizing the model score to achieve the recommended ratio of features to test points.

Table 1 Overall transformation comparison

Property	FFT	STFT	WELCH	CWT	DWT	EMD/HHT
Temporal resolution	–	+ +	–	+ +	+ +	+ +
Spectral resolution at low frequencies	+ + +	+ +	+ +	+ +	–	+ +
Spectral resolution at high frequencies	+ + +	+ +	+ +	–	–	–
Type of spectral resolution	Linear	Linear	Linear	Log	Log	Quasi-log
Interpretability	+ + +	+ + +	+ +	+	+	–
Smoothening	–	–	+ +	–	+	–
Live application	+ +	+ + +	+ + +	–	+ + +	–

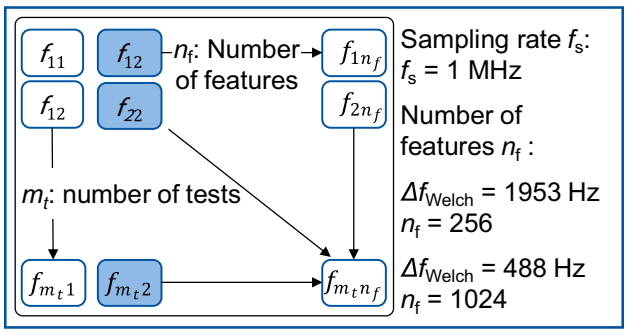

Fig. 8 Feature matrix

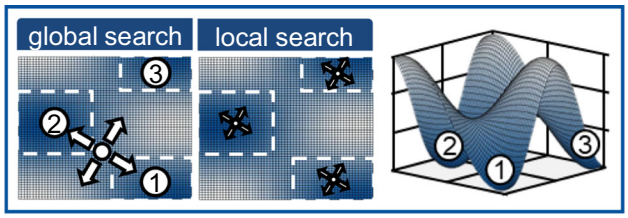

Fig. 9 Search strategies of metaheuristics

Feature selection

Feature selection is a step in the data preparation process that involves removing irrelevant or redundant features to improve model performance and prevent overfitting. There are two main classes of feature selection methods. Filter methods evaluate feature relevance independently of subsequent machine learning models. These methods use statistical coefficients, such as correlation coefficients, chi-square tests and mutual information to determine relevance. The features are ranked according to these coefficients and the best ones are selected for model training. Although filter methods are fast and simple, they do not consider interactions between features. Wrapper methods, on the other hand, evaluate features using a machine learning model. These methods test different combinations of features and select the combination that produces the best-performing model (e.g., using cross-validation). Well-known wrapper methods include forward selection, backward elimination and recursive feature elimination. Wrappers are usually more accurate, but also more computationally intensive. (Jovic et al., 2015).

In this example, with a large number of features, classic wrapper methods cannot be used due to excessive computing times, or they can only be used with great effort (e.g., use of computing clusters). To overcome this challenge, while at the same time taking interactions between features into account, a feature selection method combining a wrapper method and optimization algorithms was developed to reduce computing time. In this method, selecting the best features is formulated as an optimization problem that aims to maximize the score of a machine learning model.

There are two basic methods for solving optimization problems: exact methods and heuristics. Unlike exact methods, heuristics do not calculate exact solutions, rather, they approximate the optimum (Sörensen, 2015). Most heuristics use random variables for the approximation procedure. These stochastic calculation steps lead to the disadvantage that the results are not reproducible. However, they make it possible to find the best global solution. Using process-specific parameters to terminate the optimization method significantly reduces the computing time of heuristics compared to exact methods. Additionally, a compromise can be found between solution exactness and computing time. A subgroup of heuristics are metaheuristics. Metaheuristics can be classified as local, constructive, population-based, or trajectory-based methods. Metaheuristics are developed and applied to address the contradictory requirements of rapidly examining the entire search space for areas with high-quality values (diversification) and determining the optimum in local areas as accurately as possible (intensification) (Blum & Roli, 2003). Figure 9 illustrates the strategic possibilities of a metaheuristic in the global and local searches for minimums.

The presented architecture uses a Genetic Algorithm (GA) and a Particle Swarm Optimization (PSO) as its optimization algorithms. Both of these algorithms are classified as population-based. The main difference between these two

metaheuristics lies in their application and the implementation of their global and local search strategies. First, the GA restricts the search space to areas with high-quality values, then searches for the global optimum within these areas. PSO searches various local best solution spaces and extends the search to the entire search space. For the optimization, the GA was used for global search and PSO for local fine-tuning. The algorithms were implemented for two main reasons: first, both search algorithms can be applied efficiently even in high-dimensional spaces (in this case, the number of features) and are well suited for parallelization. Second, the frequency spectra derived from the time series data are characterized by abrupt changes and a high number of local optima. Consequently, the feature-extracted search space can be considered 'rough'. In this context, the GA offers advantages over other optimization algorithms, e.g. a Bayesian Optimization (Dalio et al., 2017). To reap the benefits of both the GA and PSO, they were combined and managed within the presented architecture using a higher-level optimization method. This method is described below, followed by a detailed presentation of the individual algorithms.

Structure and procedure of the hyperheuristic

Based on previous research in this field, the name hyperheuristic was chosen for the term of the superordinate optimization method (Burke et al., 2013). In the case presented here, the hyperheuristic maps the following tasks:

- Pre-processing of the feature matrix from chapter 2
- Definition of the number of features for the subsequent feature selection and definition of the search space
- Initialization of the individual optimization algorithms
- Receipt of the determined solutions of the individual optimizations and, if necessary, adjustment of the individual optimizations

Figure 10 shows both the individual functions of the hyperheuristic and the optimization algorithms within a diagram. The hyperheuristic operates at the optimization control level, whereas the individual algorithms are implemented at the subordinate optimization level. Derived from the hyperheuristic tasks described above, the hyperheuristic functions were essentially implemented for preprocessing and optimization control. Within a preprocessing step, an imputation is performed, whereby missing or erroneous entries from the feature matrix from Chapter 2 are replaced by an interpolation of the neighboring points. In addition, the feature matrix is scaled according to the form

$$z = \frac{x - \mu}{\sigma_f} \tag{25}$$

where z stands for the standardized value of the feature, which is calculated from the original value x and the mean value μ and the standard deviation σ_f of the feature (Hastie et al., 2017). Standardization scales each feature to a mean of zero and a standard deviation of one, which prevents the dominance of features with larger numerical values in the initial state.

A parameter list is created based on the pre-processed feature matrix in the pre-selection state. This list defines the search space for all features and serves as the basis for creating the first optimization iteration for the optimization algorithms later on. The number of rows in the parameter list corresponds to the number of features to be selected, which is determined by the person applying the optimization. For each row, a segment and a lower and upper frequency limit are entered column by column. Next, the optimization algorithms are initialized. In this method, a new worker is started on a new computing core for each optimization algorithm to enable parallel processing. During initialization, the algorithms are given the parameter list, a termination criterion for the optimization, the pre-processed feature list, and the associated targets. To enable communication between the algorithms and hyperheuristic, queues are set up in the architecture.

The optimization algorithms develop solutions based on the hyperheuristic's input parameters. In this case, the possible solutions are the feature matrices used to train and validate machine learning models. The feature matrix that achieves the best machine learning model score is considered the best solution for the optimization algorithm. Within an optimization algorithm, a new best solution is determined within every iteration and forwarded to the hyperheuristic via the queue. The hyperheuristic then checks whether the optimization algorithm requires adjustments to its optimization procedure. Four different cases can occur:

- Case 1: A defined minimum number of iterations within the optimization algorithm has not yet been reached.
- Case 2: The minimum number of iterations has been reached and the optimization algorithm sends the globally best solution.
- Case 3: The minimum number of iterations has been reached and the optimization algorithm has sent an algorithm-related best solution.
- Case 4: The minimum number of iterations has been reached and the optimization algorithm has not sent a solution improvement for a defined number of iterations.

Since metaheuristics generally require a minimum number of iterations and temporary non-improvement of solutions is possible, a variable was set up to monitor these states for cases 1 and 4. The global best solution refers to the best

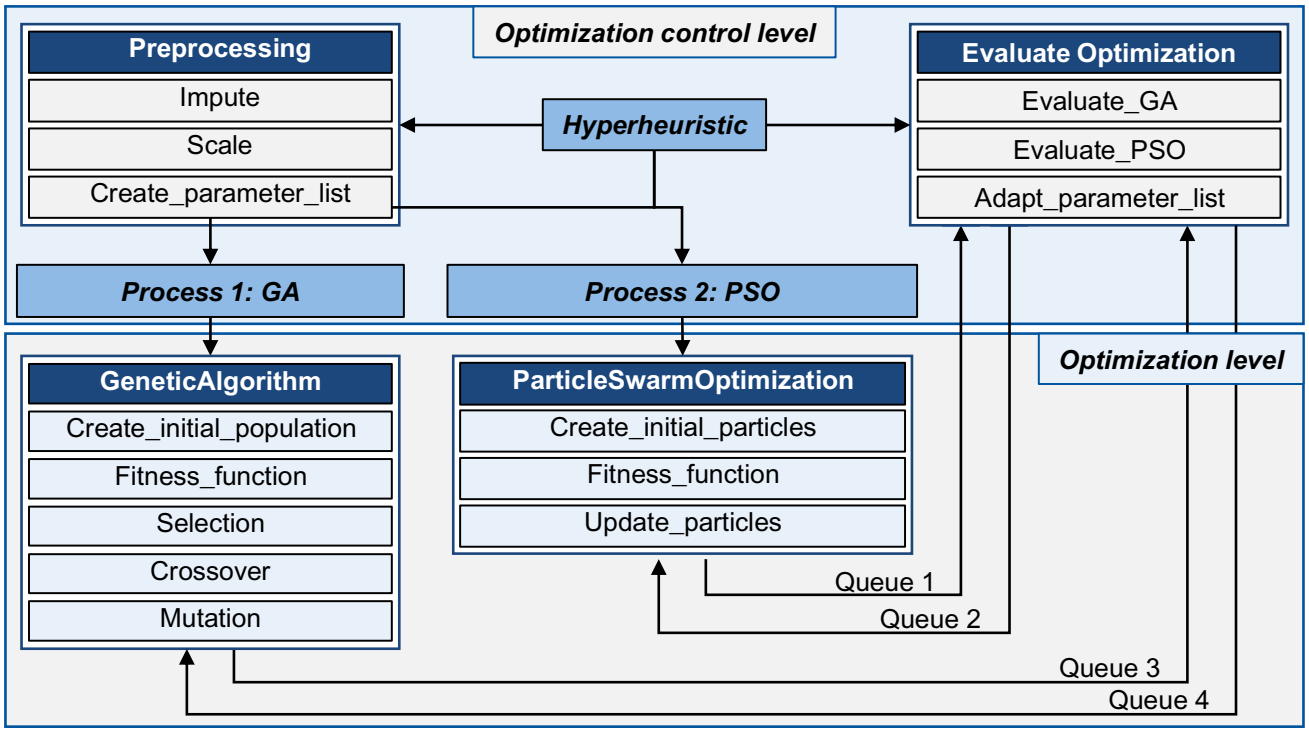

Fig. 10 Optimization architecture for feature selection

solution across all algorithms. In cases 1 and 2, the hyperheuristic does not initiate any search adaptation. In case 3, the current global best solution is transmitted to the optimization algorithm under consideration. In case 4, a new parameter list is sent to the algorithm, which is then restarted with new search space limits. This new parameter list contains a search space adjustment for each feature in the feature matrix, based on the previous global best solution. To achieve this, a bandwidth b_w is added to and subtracted from each frequency of the previous global best solution, thus defining new search space limits. Based on a minimum and maximum frequency f_{min} and f_{max}, the bandwidth b_w is calculated with

$$b_w = \frac{(f_{max} - f_{min}) \cdot c_B \cdot c_i}{2} \tag{26}$$

where c_b is a user-specific but constant coefficient for setting a basic bandwidth and c_i is the reciprocal of the number of iterations. As the number of iterations increases, c_i decreases, whereby the bandwidth b_w becomes smaller with each optimization adjustment. This results in a reduction of the search space, which reinforces the intensification effect described above.

Figure 11 illustrates the chronological sequence of the optimization cycle in the hyperheuristic, as explained previously. When the hyperheuristic is initiated, the overall optimization is established by calling the feature matrix and the targets. This is followed by preprocessing through imputation and scaling. After the parameter list is created, the GA and PSO begin. Within the optimization loop, the queues of the individual algorithms are queried repeatedly for new solutions.

When a new solution reaches the hyperheuristic, the system checks which of the four upper cases has occurred and if the optimization needs adjustment. If an adjustment is not necessary, the next queue is queried for a new solution. If an optimization adjustment is necessary, either the globally best solution is forwarded, or a new parameter list is sent. When a termination criterion is met within an optimization algorithm, the hyperheuristic is notified via the queue. The overall optimization ends when all the algorithms reach their termination criterion.

Structure and procedure of the genetic algorithm

The GA's optimization strategy is based on the principles of evolution. It includes functions for forming a population, selecting the best individuals within that population and recombining and mutating genetic material. Each of these principles corresponds to a step in the GA (Lambora et al., 2019). GAs are widely implemented for optimization problems and are usually carried out according to the listed steps, with various adjustments made within the individual steps depending on the application.

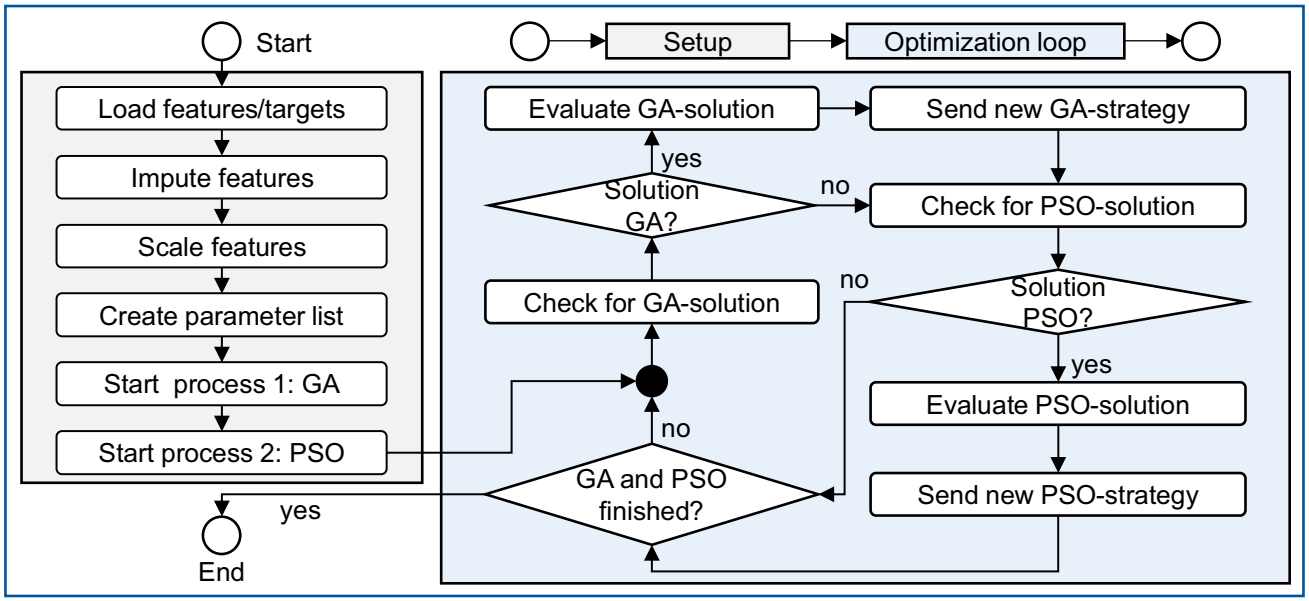

Fig. 11 Optimization cycle of the hyperheuristic

In the present application, a population is formed from a defined number of feature matrices whereby one individual in the population corresponds to one feature matrix. Recombination of genetic material is achieved through column-wise recombination of different features of two feature matrices. Mutation involves replacing a single feature.

Figure 12 illustrates the fundamental functions of the GA's pseudo code in this architecture. Object-oriented programming with a GeneticAlgorithm class was used to implement the GA. An object of this class is initialized by providing the population size, number of generations, mutation rate, feature matrix, targets, and parameter list. In this case, the population size corresponds to the number of feature matrices with the final selected features, as well as the number of machine learning models trained and validated during a generation of the algorithm. Process result variables from the grinding process, such as roughness or edge zone damage, serve as targets. Additionally, the object receives the feature matrix in its pre-selection state and the aforementioned parameter list. The number of generations represents the termination criterion of the optimization algorithm, i.e., the required number of iterations until termination.

The first step of the GA is to form a starting population. With a population size of 100, for example, 100 feature matrices are formed. Each feature matrix is considered an individual. To create an individual, feature IDs consisting of segment names and frequencies are randomly generated based on the parameter list. If a feature matrix consists of ten features, ten feature IDs are randomly generated. These IDs are then used to select features from the feature matrix transferred by the hyperheuristic. Based on the starting population, the fitness of all created feature matrices is evaluated using a fitness function. This function represents the training and validation of a machine learning model. No special boundary conditions need to be met to select a machine learning model. Based on a training and test dataset, only a cross-validation score, such as the coefficient of determination R^2 or root mean squared error RMSE, needs to be calculated. Assigning a score to each feature matrix in a population allows to rank the best solutions and pass the best solution on to the hyperheuristic. In the next selection step, the best feature matrices are selected and used for recombination. The architecture presented implements a tournament mode for selection, in which three individuals from the original population are compared in several runs based on their fitness (model score) and the individual with the highest score is selected for recombination. During recombination, two feature matrices are randomly selected and their features are recombined. This creates two new feature matrices, which are added to a new population. At random points, individual features are replaced with others of the population within the mutation with a probability equal to the specified mutation rate. These steps are repeated until the termination criterion is met, completing the optimization by the genetic algorithm. Figure 13 shows the time sequence of the implemented GA. In addition to the pseudocode explained above, the diagram shows the solution exchange with the hyperheuristic, as well as the optimization adaptation. In the architecture, the best solution is sent to the hyperheuristic after each generation. After sending the solution, a check is made to see if the termination criterion has been met. If not, the hyperheuristic retrieves the new strategy.

```
Class GeneticAlgorithm(population_size, generations, mutation_rate, feature_matrix, targets, parameter_list):
  Function run():
    population ← create_initial_population()
    FOR each generation in generations:
      FOR each individual in population:
        fitness ← fitness_function(individual)
      best_individual ← individual with best fitness
      save (fitness of best_individual)
      selected_population ← selection(population, fitnesses)
      FOR each parent (parent1, parent2) in selected_population:
        (child1, child2) ← crossover(parent1, parent2)
        child1 ← mutation(child1)
        child2 ← mutation(child2)
        next_population add: child1, child2
  Function create_initial_population():
    FOR i in 1 to population_size:
      FOR each (start, end) in parameter_list:
        Select random feature-ID between start and end
        Add feature to feature_list
        individual ← Create initial dataframe
                     from feature_list and feature_matrix
        Add individual to population
    RETURN population
  Function fitness_function(individual):
    model ← Select machine learning model
    Split individual and targets into training and testdata
    fitness ← Calculate Cross-Validation-Score (e. g. R²)
    RETURN fitness
  Function selection(population, fitnesses):
    FOR i in 1 to population_size:
      Select random individuals + fitnesses
      winnner ← individual with highest fitness
      Add winner to selected_population
    RETURN selected_population
  Function crossover(parent1, parent2):
    Combine columns of both partents to pool
    child1 ← Rand. selection of  columns (size like parent1)
    child2 ← Rand. selection of  columns (size like parent2)
    RETURN (child1, child2)
  Function mutation(individual):
    FOR each column in individual:
      IF random_number < mutation_rate:
        Replace column with new column
    RETURN individual
```

Fig. 12 Pseudocode of implemented Genetic Algorithm

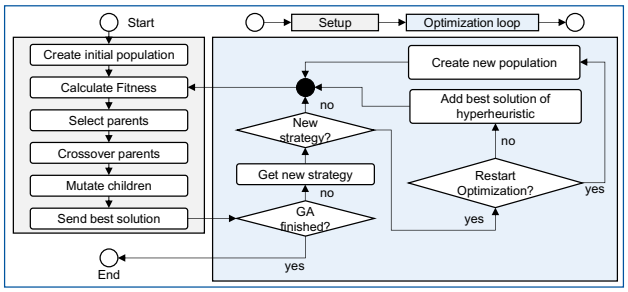

Fig. 13 Optimization cycle of the Genetic Algorithm

If no adjustments are necessary based on the optimization progress, the fitness values of the current generation are determined, and the next steps are carried out based on them. If a new strategy is used, either the best solution of the current generation is added or a new optimization is started. To this end, the hyperheuristic generates a new starting population based on the newly transmitted parameter list.

The three main optimization parameters of the GA are population size, number of generations, and mutation rate. A larger population size provides better coverage of the entire search space and a higher probability of finding good solutions. However, as the population size increases, so does the computational effort required. Increasing the mutation rate maintains genetic diversity and avoids early convergence. Conversely, a mutation rate that is set too high can prevent convergence.

Structure and procedure of the particle swarm optimization

Particle swarm optimization is an evolutionary optimization algorithm inspired by the collective movement of animals in nature, such as schools of fish or flocks of birds. Similar to GAs, PSO aims to determine the highest-quality solution within a defined search space. The first step in the optimization process involves creating initial particles, each of which is randomly placed within the search space at a position x and given an initial velocity v. Moving the particles through the search space at velocity v achieves iterative improvement of the global solution. The movement of the particles is controlled by the best solution of each particle and the best solution across all particles. With regard to the implemented feature selection method, a particle corresponds to a feature matrix and a solution. The position of a particle corresponds to the frequencies of a feature matrix. As with the GA, the quality of a feature matrix is specified by the model score of a machine learning model. For each iteration within the optimization algorithm, the speed of a particle v is determined by the formula

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (p_i - x_i(t)) + c_2 \cdot r_2 \cdot (g - x_i(t)) \tag{27}$$

which consists of three terms (Wang et al., 2018). The first term uses the inertia weight w to determine the extent to which the current velocity of particle i at time t should be

```
Class PSO(particle_size, iterations, std_dev, c1, c2, w, feature_matrix, targets, parameter_list):
  Function run():
    particles ← create_particles()
    velocities ← create_velocity_list with values of std_dev
    dataframes ← create_dataframe_list(particles)
    fitnesses ← fitness_function(particles)
    personal_bests ← particles
    global_best ← Find best particle
    FOR each iteration in iterations:
      fitnesses ← fitness_function(dataframes)
      personal_bests ← Update personal best positions
      global_best ← Find best global particle
      particles, velocities ← update_particles(particles,
                  personal_bests, global_best, velocities)
      dataframes ← create_dataframe_list(particles)
  Function create_particles():
    FOR each particle in particle_size:
      FOR each segment in segment_list:
        feature_list ← create_feature_list()
        Add feature_list to segment_dict
        Add segment_dict to particle
    RETURN particles
  Function create_feature_list():
    FOR each parameter in parameter_list for one segment:
      feature ← random value within bounds
      feature_list ← Add feature to feature_list
    RETURN feature_list
  Function create_dataframe_list(particles):
    FOR each particle:
      FOR each segment in particle:
        Select columns for features from feature_matrix
        Rename and merge into dataframe
      Add dataframe to dataframes
    RETURN dataframes
  Function fitness_function(dataframes):
    model ← Select machine learning model
    FOR dataframe in dataframes:
      Split dataframe and targets into training and testdata
      fitness ← Calculate Cross-Validation-Score (e. g. R²)
      Add fitness to fitnesses
    RETURN fitnesses
  Function update_particles(particles,personal_bests,
  global_best,velocities):
    Compute new velocities based on PSO-parameters
    Apply velocities to position
    RETURN particles, velocities
```

Fig. 14 Pseudocode of implemented PSO

maintained for time $t + 1$. The second term defines the extent to which the velocity changes on the basis of the personal best solution p_i. The third term expresses how the velocity of a particle is changed on the basis of the globally best solution g. Terms one and two can be adjusted by different weights via the acceleration coefficients c_1 and c_2. By integrating two random variables r_1 and r_2, a stochastic movement of the particles in the search space is also taken into account. Figure 14 shows the pseudocode of the implemented algorithm.

Similar to the GA, a PSO class was implemented that contains the following transfer parameters: particle_size, iterations, std_dev, c1, c2, w, feature_matrix, targets, and parameter_list. The number of particles or the number of subsequent feature matrices is determined by particle_size. The termination condition of the algorithm is represented by the number of iterations. The standard deviation std_dev is necessary for initializing the particles. The parameters c1, c2, and w correspond to the explained coefficients c_1, c_2, and w.

In the first step, the create_particle function generates particles consisting of segments and features. The features are formed from the transferred parameter list and correspond to the frequencies according to WELCH. Based on the created particles, velocities are randomly generated for the particles using the specified standard deviation. Next, feature matrices are formed based on the particles and passed to the fitness function. Assigning a machine learning score to each feature matrix determines the best current solution for each particle, and the best global solution is saved. Then, based on the model scores, the speed of the particles is updated according to formula 27, and the steps are repeated until the termination criterion is met. Similar to Fig. 14, the pseudocode is supplemented with the option of sending solutions to the hyperheuristic and receiving adjustments from the optimization strategy.

In summary, the implemented procedure offers various possibilities for further development, particularly through the parallelization of optimization algorithms. First, complementary optimization algorithms can be added. Second, the same optimization algorithms can be used with different optimization parameters.

Modeling, analysis and optimization

The presented architecture supports the training and validation of machine-learning models with a particular focus on supervised learning approaches based on systematic feature extraction and feature selection. Both classification and regression models can be implemented and are accessed within the proposed methodology using established Python libraries (Pedregosa et al., 2011). Because of the type of feature extraction, it is important to optimize the frequency resolution Δf that improves model quality the

most for each model. Regarding feature selection, the presented architecture enables an optimization of the number of features n_f and maximum frequencies f_{max}. Reducing the maximum frequency directly lowers the required sampling rate and memory demand, thereby improving computational efficiency. Beyond optimizing feature dimensionality and sampling requirements, the method further allows for a comparative evaluation of different signal sources or process segments. Their relative importance within the model can be assessed by analyzing the proportion of features originating from each signal source or segment that are ultimately selected. In addition, quantitative parameters such as the SHAPLEY value offer the possibility of evaluating the importance of an individual feature. The SHAPLEY value originates from game theory and distributes the "winnings" of a game fairly among the players by calculating the average marginal contribution of a player for all possible coalitions. Applied to machine learning, it measures the average contribution of a feature to the model prediction, providing a fair and consistent definition of feature importance (Lundberg & Lee, 2017). In terms of model quality assessment, all common parameters such as the coefficient of determination R^2 or the root mean squared error *RMSE* for regressions and the accuracy *P* for classification models can be used. Five-fold cross-validation is used as standard. Here, the data set is divided into five equal subsets ("folds"). Model building is carried out in five runs, with four folds used for training and the remaining fold used for testing in each run. At the end, the five test results are averaged (James et al., 2021). Regardless of the default setting, the proposed architecture is independent of a specific validation strategy and supports the implementation of arbitrary cross-validation schemes. In this work, both a randomized fivefold cross-validation and a blocked cross-validation were evaluated to assess model performance under different data partitioning assumptions.

Application of the architecture on a grinding process

The presented architecture is applicable to a wide range of grinding processes, including cylindrical, surface and tool grinding and can be employed to predict various result variables such as surface roughness and microstructural conditions. To achieve optimal model performance, however, application-specific tuning of feature extraction parameters (e.g., frequency resolution Δf), model parameters (e.g., number of selected features n_f) and optimization parameters (e.g., number of iterations n_g) is required. To demonstrate the suitability of the architecture, a case study is presented that illustrates the step-by-step optimization process for developing a machine-learning model intended for live operation. To approximate an industrially relevant scenario, grinding burn monitoring is investigated under varying grinding wheel manufacturers and different grinding wheel batches.

Table 2 Grinding wheel overview

Name	Manufacturer	Batch	Number	YOUNG's Modulus *E* [GPa]
GW 1	1	1	1	40.55
GW 2	1	1	2	44.25
GW 3	1	2	1	46.60
GW 4	1	2	2	50.81
GW 5	2	1	1	41.22
GW 6	2	1	2	45.68
GW 7	2	2	1	45.32
GW 8	2	2	2	46.55
GW 9	3	1	1	36,26
GW 10	3	1	2	37.31
GW 11	3	2	1	33.09
GW 12	3	2	2	35.52

Test materials and method

The tests to validate the presented method were carried out using surface grinding tests on the PROFIMAT MT 608 surface and profile grinding machine from BLOHM. A wear workpiece (dimensions: 19 × 200 × 300 mm) and a measuring workpiece (dimensions: 19 × 80 × 300 mm) made of 100Cr6 with a hardness of $H = 60$ HRC were used for the investigations. The measuring component was clamped using a test rig manufactured at the MANUFACTURING TECHNOLOGY INSTITUTE – MTI. This included a force measurement platform and the option of machining workpieces with integrated thermocouples. The wear workpiece was clamped directly onto the magnetic clamping table of the machine tool. On the tool side, 12 vitrified-bonded corundum grinding wheels (*GW 1 – GW 12*) with a mesh size of F80, a hardness grade of J and a porosity of 5 according to the manufacturer were used. The grinding wheel manufacturers were varied in three stages and the tool batches in two stages. Two grinding wheels were used per grinding wheel batch (see Table 2). In order to quantitatively represent the differences between manufacturers and batches, the YOUNG's modulus of the grinding wheels was measured using an RFDA measurement.

The cooling lubricant used was HOCUT 5050 emulsion from QUAKER HOUGHTON with a concentration of $c = 6\%$ oil content. A DDS form roller with the specification 305DS71P-150–1-2 from SAINT- GOBAIN was used for the dressing process.

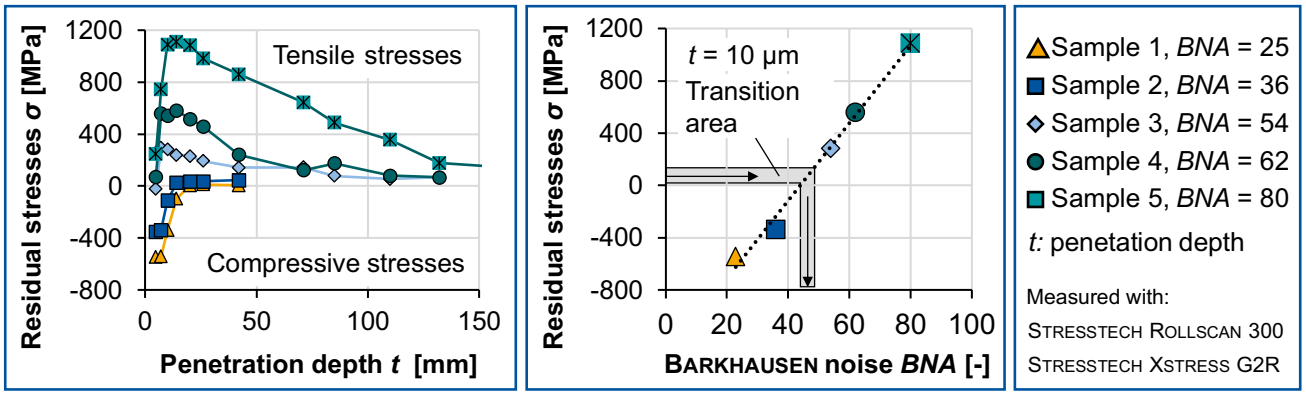

Fig. 15 XRD measurements and determination of the critical grinding burn range

The experimental investigations were conducted as tool life tests, in which each grinding wheel was used until a defined specific material removal volume of $V'_w = 1{,}600$ mm^3/mm was reached. In total, twelve test series were performed, corresponding to the use of twelve individual grinding wheels. Each test series comprised 16 test points, with each test point was defined by an incremental specific removal volume of $V'_w = 100$ mm^3/mm. The specific removal volume per test point was achieved through 50 grinding strokes on a wear workpiece with a depth of cut $a_e = 0.01$ mm, followed by six grinding strokes on a measuring workpiece using the same depth of cut a_e. Process variables were only recorded during the grinding of the measuring workpiece and included the normal and tangential grinding forces F_n and F_t, the grinding temperature T and the acoustic emission signal U_{AE}. To ensure statistical reliability, all test series were executed twice. At the beginning of each test series, the grinding wheels were dressed at a peripheral speed of $v_{s,d} = 40$ m/s, a dressing speed ratio of $q_d = 0.8$ and a dressing overlap ratio of $U_d = 2$. During the experiments, the grinding wheel speed was kept constant at $v_s = 40$ m/s, the workpiece feed speed at $v_w = 10$ m/min and the coolant outlet speed at $v_{coolant} = 28$ m/s.

As a metric for evaluating thermally induced tool damage caused by the grinding process, a BARKHAUSEN noise analysis was performed on the ground workpieces. For this purpose, the BARKHAUSEN noise analyzer Rollscan 300 and an S1-164–15-08 sensor from the manufacturer STRESSTECH were used. The microstructural condition was recorded at a magnetizing voltage of $U_{Mag} = 4.0$ V and a magnetizing frequency in the form of a sinusoidal wave of $f_{Mag} = 120$ Hz. To determine the transition into regions affected by grinding burn, X-ray diffraction (XRD) measurements were additionally carried out using the X-ray diffractometer STRESSTECH XSTRESS G2R. For this purpose, near-surface residual stresses were first determined at a maximum penetration depth of $t = 200$ µm on five ground component samples exhibiting varying BARKHAUSEN noise amplitudes *BNA* in the range from $BNA_{min} = 25$ to $BNA_{max} = 80$, measured in both the longitudinal and transverse directions. Figure 15 shows the residual stress profiles σ as a function of the component samples and the penetration depth t. An increase in residual stresses with rising BARKHAUSEN noise amplitudes *BNA* can be observed, with a transition from compressive to tensile stresses occurring at component specimen 3 at a value of $BNA = 55$.

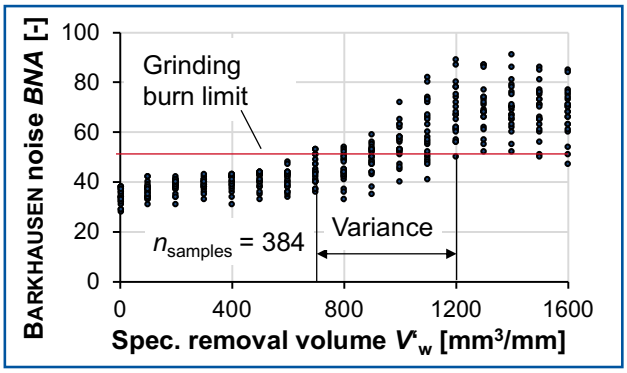

Fig. 16 Results of the BARKHAUSEN noise

Following the XRD measurements, a correlation was established between the measured BARKHAUSEN noise amplitudes and the determined residual stresses at a penetration depth of $t = 10$ µm. By means of linear interpolation between the component specimens, a transition range of the critical BARKHAUSEN noise amplitude *BNA* was identified, above which grinding burn is highly likely to occur. Based on these results, the transition range was defined between a minimum BARKHAUSEN noise amplitude of $BNA_{min} = 45$ and a maximum BARKHAUSEN noise amplitude of $BNA_{max} = 50$.

Figure 16 provides a general overview of the measured BARKHAUSEN noise amplitudes *BNA* of all grinding wheels as a function of the specific material removal volume V_w'. In addition, the defined transition range to workpieces affected

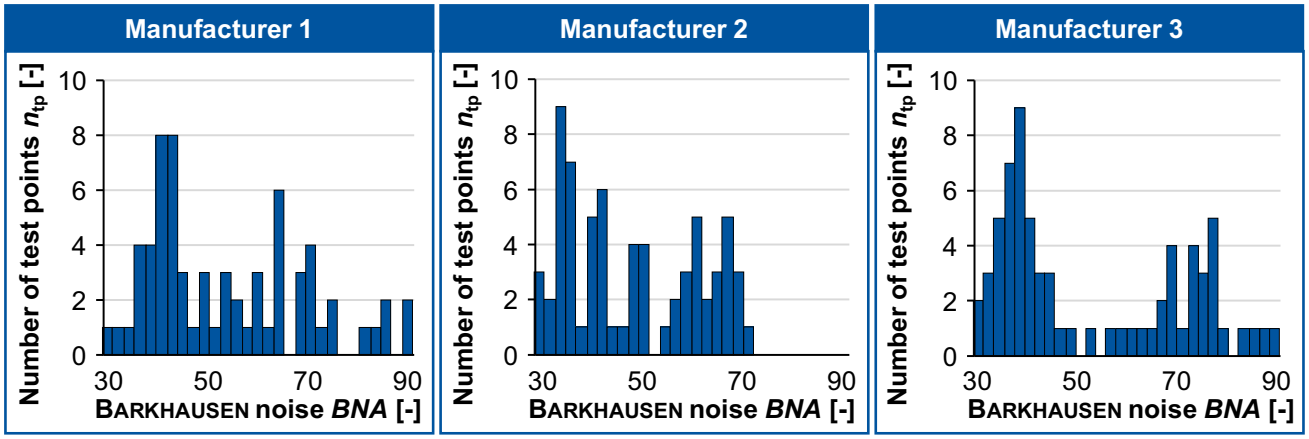

Fig. 17 Distribution of BARKHAUSEN noise measurements across manufacturers

by grinding burn is indicated. In general, a qualitatively comparable trend of the BARKHAUSEN noise amplitude *BNA* was observed for all grinding wheels. Initially, a nearly constant behavior was identified, which persisted up to an average specific material removal volume of $V'_w = 700$ mm^3/mm. This constant region was followed by a pronounced increase in the BARKHAUSEN noise amplitude within the range of 800 mm^3/mm $\leq V'_w <$ 1,200 mm^3/mm. Finally, a decrease in *BNA* was observed up to the defined material removal volume of $V'_w = 1{,}600$ mm^3/mm. Due to the fact that the grinding burn limit falls on different removal volumes due to the variances, monitoring of the BARKHAUSEN noise is necessary in industrial applications. In addition to the overall view, Fig. 17 shows the histograms of the measured values for all manufacturers. The three histograms show noticeable differences in the distribution and spread of the BARKHAUSEN noise amplitudes *BNA* for the respective manufacturers. Overall, manufacturer 1 tended toward lower and more narrowly distributed *BNA*-values, manufacturer 2 showed a structured distribution with two dominant ranges and manufacturer 3 exhibited the broadest distribution with an increased frequency of higher *BNA* values.

Data acquisition and modeling

The BLOHM PROFIMAT MT 608 profile and surface grinding machine available at the Manufacturing Technology Institute—MTI was used for process monitoring. Figure 18 shows the infrastructure for recording machine control and sensor data, as described in Chapter 2. TYROLIT TOOLSCOPE was used to record the machine control data. On the sensor side, the acoustic emission signal was recorded with a KISTLER sensor of type 8152C. A 3-component dynamometer (KISTLER 9255C) and a device for measuring temperature and cooling lubricant were also installed but were not used in this case study.

To validate the method, the focus was set on the acoustic emission signal U_{AE} and was recorded at a sampling rate of $f_{s,AE} = 800$ kHz. In addition, the machine control data was recorded at a sampling rate of $f_{s,C} = 100$ Hz. By performing the grinding tests as pendulum grinding processes, one stroke was performed in upgrinding and one stroke in downgrinding alternately. The data was stored in TDMS format, with each file having a size of approximately 150 MB. After the tests, the grinding strokes were segmented in up and downgrinding based on the speed of the x-axis of the grinding machine. The last two strokes of the grinding processes were transferred to feature extraction as grinding segments.

In order to obtain an optimized model for economical grinding burn monitoring, the model selection, frequency resolution Δf, number of features n_f and sampling rate $f_{s,AE}$ were optimized in the modeling step. For this purpose, the listed parameters were varied step by step and the results were evaluated. The features included in the final model were then analyzed with the optimized model in terms of their segment and frequency. Particular attention was paid to using different model types on the model side. These included a type of linear regression (Bayesian Ridge Regression: BR), a kernel-based method (Support Vector Regression: SVR) an instance-based method (k-Nearest Neighbors: kNN) and a tree-based method (LightGradientBoosting: LGBM). To approximate industrial applicability and to prevent data leakage, a blocked cross-validation (blocked CV) with two splits was applied first (see Fig. 19).

For each split, the models were trained using the experimental series of nine grinding wheels and validated using the remaining three grinding wheels. In both training blocks, grinding wheels with both the highest and the lowest YOUNG's modulus E from each grinding wheel manufacturer were included. The grinding wheels with a medium YOUNG's modulus were used for training in one split and for validation in the other split. Due to the limited amount

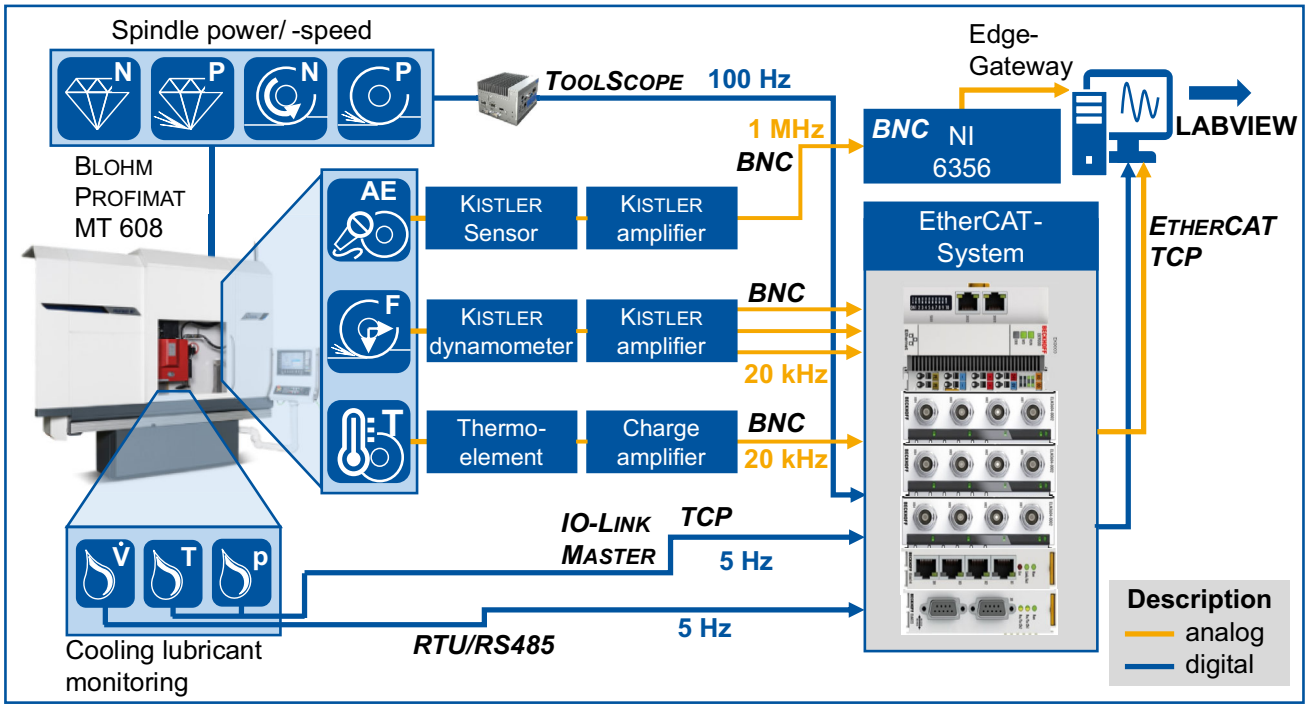

Fig. 18 Data acquisition system for surface grinding (Image source: Blohm Jung, Peter Lehmann; Tyrolit)

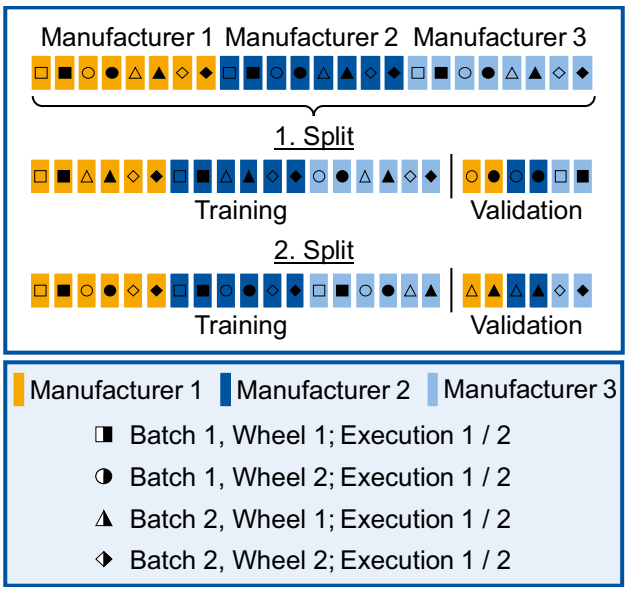

Fig. 19 Cross-validation procedure

of data, cross-validation with a higher number of folds as well as a separate test holdout was omitted. The following results therefore primarily demonstrate the methodology for optimizing the ML models and should not be considered an industrial benchmark for grinding burn prediction.

Results and analysis options

Figure 20 shows the four steps that were carried out to derive an economically optimized model. In step one, a preselection of models was made based on basic settings for the modeling and optimization algorithms. For this purpose, four models were trained and evaluated using the described cross-validation method with a score based on the coefficient of determination R^2. The figure shows the best results for step one for the the BR, LGBM and kNN models with default settings (frequency resolution of $\Delta f = 390$ Hz and a number of features of $n_f = 50$). The cross-validation scores (CV-Score R^2) achieved represent the mean value from five training and validation runs, with the error bars representing the standard deviation. Bayesian Ridge Regression achieved the highest cross-validated performance with $R^2 = 0.83$ and was therefore selected for subsequent optimization steps, outperforming SVR, kNN, and LGBM. Step two involved investigating the extent to which changing the frequency resolution Δf improved model performance. For the selected BR model, the frequency resolution Δf was varied over several orders of magnitude. The results show a clear optimum at intermediate resolutions, while both very coarse and very fine resolutions lead to reduced performance, indicating a trade-off between information loss and noise amplification. In step three, the number of selected features n_f was systematically varied while keeping $\Delta f = 390$ Hz. Starting with $n_f = 10$, it was increased to a maximum of $n_f = 100$. Model

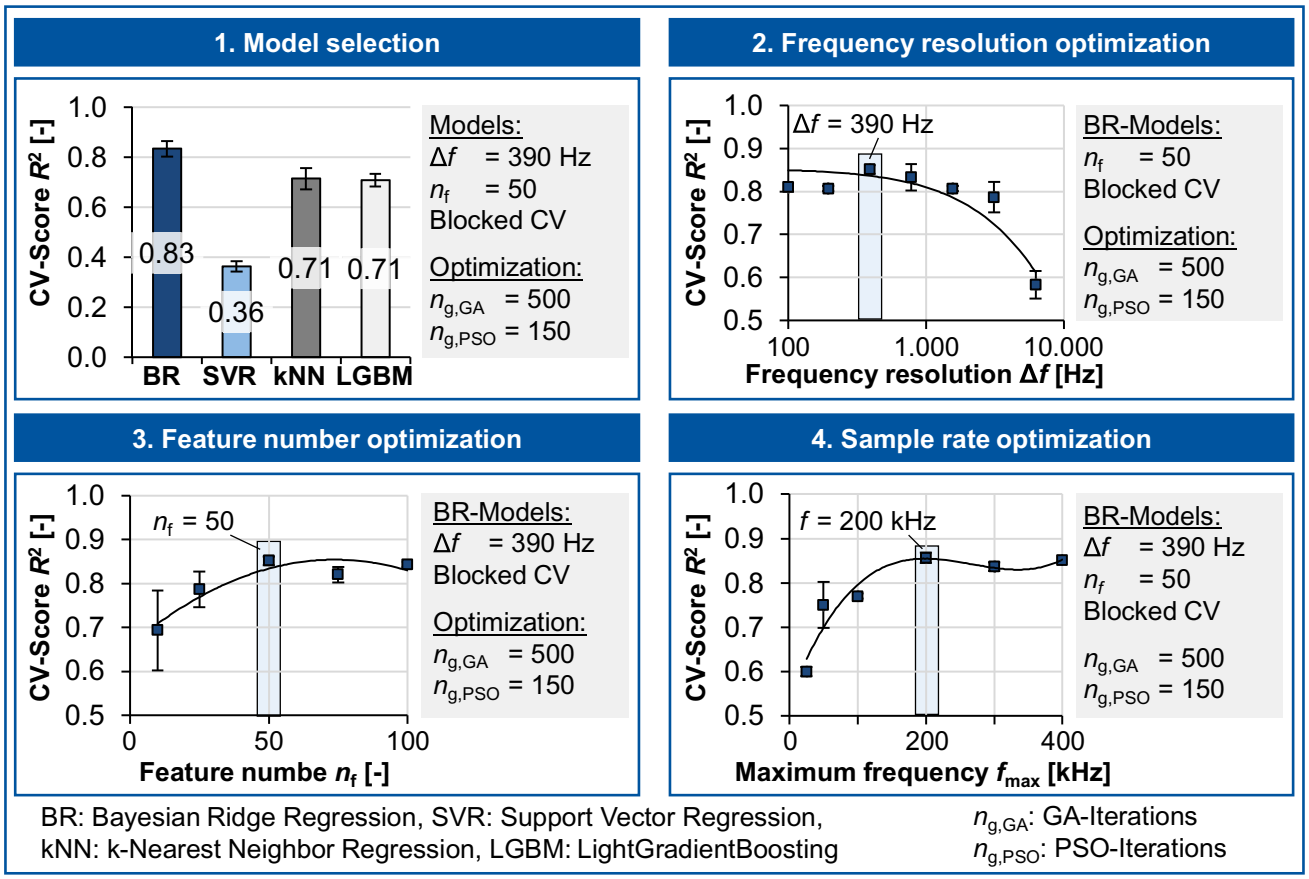

Fig. 20 Model results and optimization

performance increased with feature number up to an optimum around $n_f \approx 50$, after which performance saturated or slightly decreased, suggesting diminishing returns and potential overfitting for larger feature sets. Finally, the maximum frequency f_{max} was optimized, showing that model performance improved with increasing frequency content up to an optimum around $f_{max} = 200$ kHz, beyond which no further gains are observed. In this way, depending on the required model accuracy, the sampling rate could be reduced to $f_s = 400$ kHz taking the NYQUIST theorem into account.

Figure 21 provides a deeper insight into the model by showing the selected features of the previously best-rated model, categorised by grinding segment and frequency range. The total number of selected features differs only slighly between the two process phases, with 24 features assigned to upgrinding and 26 features to downgrinding. The plots in (b) and (c) display the selected features per grinding segment in histogram form, categorised by the frequency range. The 10 most relevant features for the model's decision-making process, as calculated using the SHAPLEY value, are shown in yellow. A representative example from the individual repetitions was used for the view.

Notably, (b) shows that a relatively large fraction of the upgrinding features was identified as important according to their SHAPLEY values, indicating that many frequency components contributed meaningfully to the model predictions in this segment. These SHAPLEY -relevant features in upgrinding were distributed across the entire frequency range, suggesting that the model relied on a broad spectral representation to capture the underlying process behavior. In contrast, (c) shows that downgrinding was characterized by fewer SHAPLEY -dominant features, which were more strongly concentrated in specific, predominantly lower-frequency bands.

To provide a performance comparison with simple filter methods for feature selection, Fig. 22 compares the performance of a BR model using randomized fivefold cross-validation and blocked cross-validation, evaluated by the coefficient of determination R^2 for training and validation. The machine-learning pipeline for the filter method applied an initial multicollinearity reduction using a variance inflation factor (*VIF*) threshold of $VIF > 5$ to remove highly correlated features. Subsequently, a univariate feature selection based on statistical significance (p-value) was performed to rank the remaining features. The top 50 features with the lowest p-values were retained and used for model training

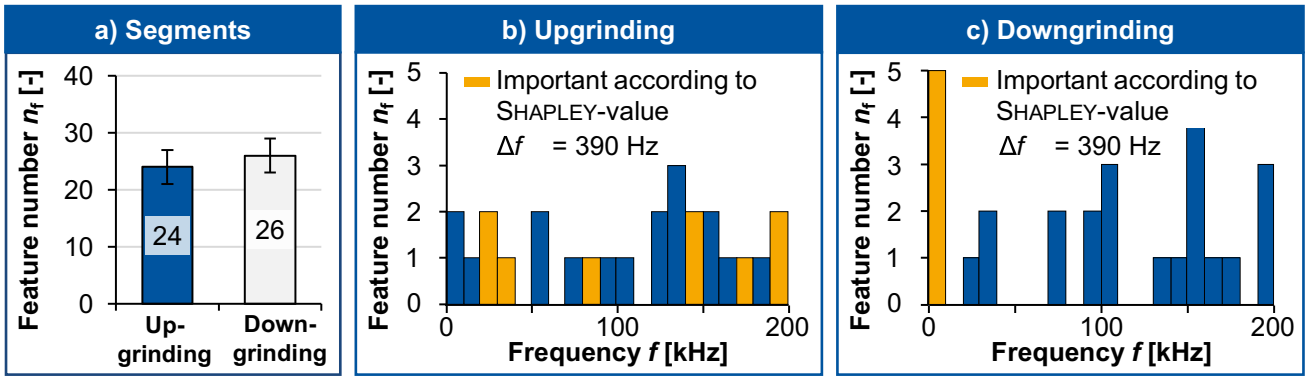

Fig. 21 Model analysis

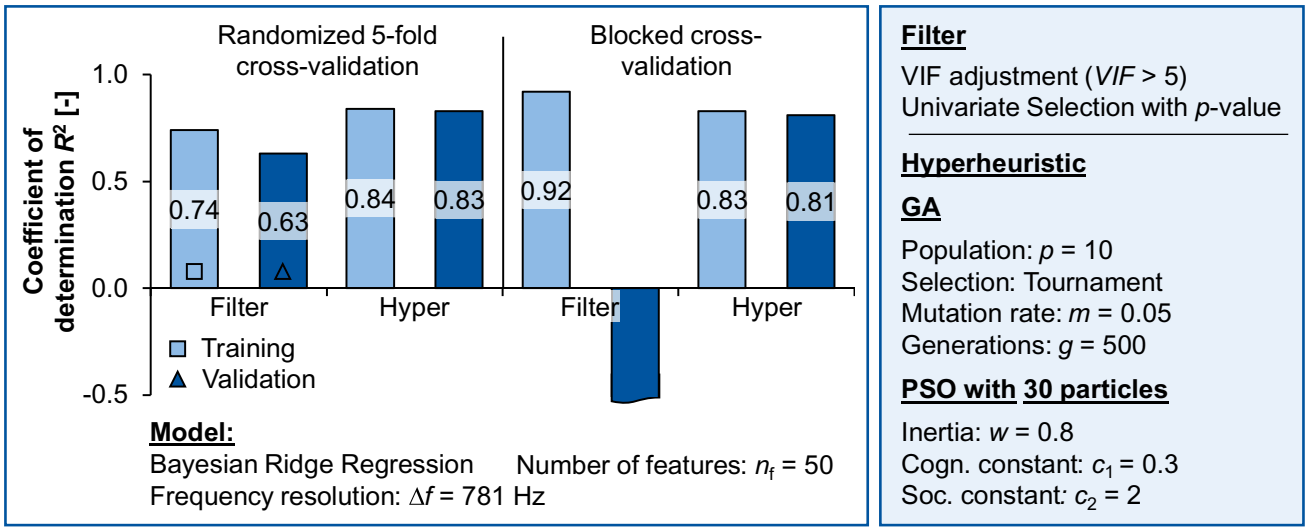

Fig. 22 Performance comparison between the hyperheuristic and filter methods

and validation. Due to the better results obtained with the filter method at a frequency resolution of $\Delta f = 781$ Hz, this was used for comparison purposes.

Under randomized fivefold cross-validation, both approaches achieved high training performance and comparatively high validation R^2 values, with the hyperheuristic approach yielding the best validation results. These results indicate an apparently good generalization when samples are randomly mixed across folds. In contrast, blocked cross-validation lead to a substantial drop in validation performance, particularly for the filter-based approach, which even exhibited negative R^2 values. Although only a filter-based feature selection method using SelectKBest with a p-value criterion was applied in this study, it represents a common class of statistical filtering approaches. Therefore, the observed limitations are highly likely transferable to other filter-based feature selection methods e.g. a RELIEF-algorithm. The results demonstrate that hyperheuristic optimization yields more generalizable models for industrially relevant scenarios with strict data separation.

Figure 23 compares the performance of a GA and the hyperheuristic approach using the coefficient of determination R^2 for different population sizes ($p = 10$ and $p = 50$) and numbers of selected features ($n_f = 10$ and $n_f = 50$), evaluated with blocked cross-validation.

For both population sizes, increasing the number of selected features from $n_f = 10$ to $n_f = 50$ resulted in a clear improvement in predictive performance. At low feature dimensionality like $n_f = 10$, both methods exhibit comparatively large error bars, indicating increased variability and reduced stability due to the limited representational capacity of the feature set. This effect was more pronounced for the GA, suggesting a higher sensitivity to initialization and stochastic effects when the search space is strongly constrained. In contrast, the hyperheuristic shows smaller error bars and more consistent performance, reflecting improved robustness under feature-limited conditions. With increasing population size p and feature number n_f, error bars decreased for both approaches, indicating more stable convergence. Both methods benefited from improved exploration due to

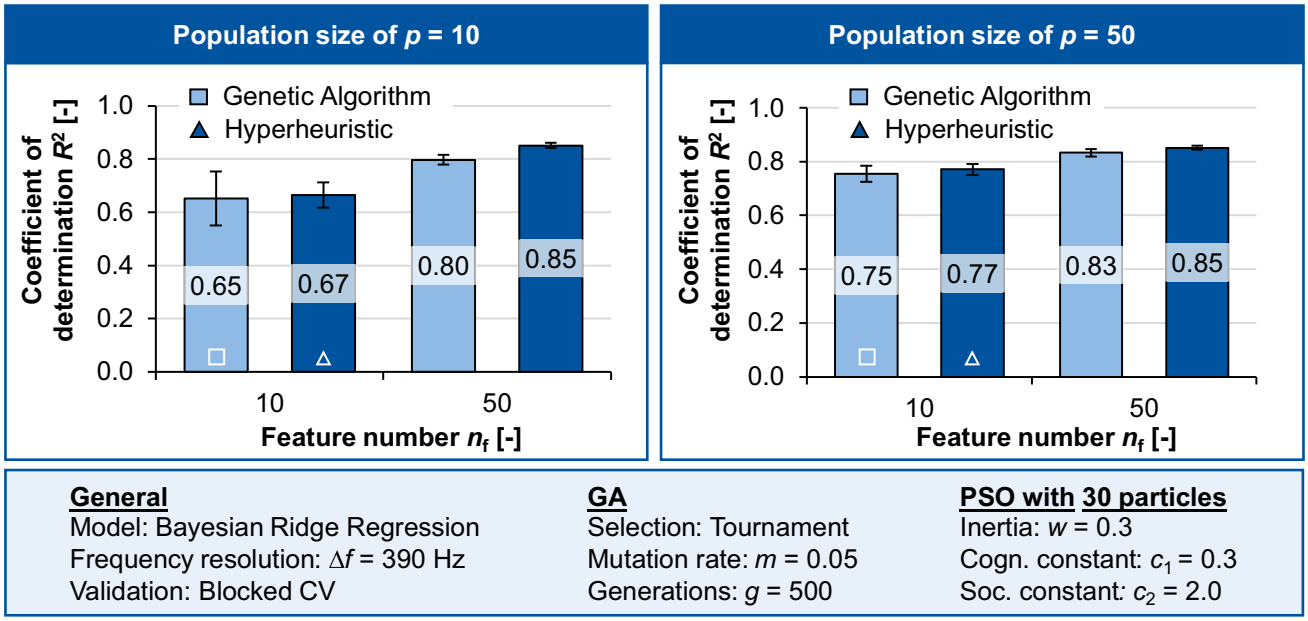

Fig. 23 Performance comparison between the hyperheuristic and a GA-Optimization

an increased population, yet the hyperheuristic maintained a systematic advantage. Overall, the results highlight that the hyperheuristic optimization not only improved mean performance but also reduced variance, which is particularly beneficial for complex or data-limited optimization problems. The advantage of the hyperheuristic is expected to become more pronounced in more complex optimization problems, as the size of the search space increases with higher feature dimensionality and stronger interdependencies between features. By adaptively combining and steering multiple optimization heuristics, the hyperheuristic can avoid premature convergence and explore the solution space more effectively than a single GA. Consequently, its robustness and stability gains are likely to scale with problem complexity, whereas single-heuristic approaches increasingly depend on careful parameter tuning and larger populations.

Finally, Table 3 compares the computational effort required for feature extraction and feature selection (FS) using the filter-based FS, a GA-based FS and a hyperheuristic-based FS with a frequency resolution of $\Delta f = 390$ Hz and varying feature set sizes ($n_f = 10$ and $n_f = 50$ features). The results were achieved with an Intel Core i7-14,700 and refer exclusively to the case study presented.

With a frequency resolution of $\Delta f = 390$ Hz, a feature extraction time of $t_{fe} = 0.049$ s was achieved. By comparison, a CWT transformation with scales of $s = 35$ required for this exact use-case a feature extraction time of $t_{fe,CWT} = 4.75$ s, which increased to $t_{fe,CWT} = 9.36$ s at a scale of $s = 70$. Filter-based feature selection showed consistently negligible computation times ($t_{filter} \leq 0.15$ s) independent of the feature number n_f, confirming its computational efficiency but also its methodological simplicity. In contrast, GA- and hyperheuristic-based feature selection required several orders of magnitude more computation time, with runtimes increasing for larger feature sets n_f. Notably, the hyperheuristic exhibited slightly higher runtimes than the GA, which is expected given its higher-level control and combination of optimization strategies.

When considered together with the previously discussed performance results, the table highlights a clear trade-off between computational cost and model quality. While filter-based methods are computationally inexpensive, they were shown to suffer from severe overfitting and poor generalization under blocked cross-validation. GA-based optimization improved performance but was less robust and scalable than the hyperheuristic. Importantly, the optimization of frequency resolution and maximum frequency enabled a targeted reduction of sample rate and therefore the optimization of storage costs without sacrificing predictive performance. Consequently, the combination of hyperheuristic feature selection with an optimized spectral representation provides a practical and economically viable pathway to achieve robust, high-performing ML models for industrial applications.

In addition to the computing times for training and validating the models, the computing times for live use were also recorded. For this purpose, a pre-trained, optimized BR model with a frequency resolution of $\Delta f = 390$ Hz and a feature number of $n_f = 50$ was used. Feature extraction was designed so that only the 50 features that were actually needed for the prediction were determined. This approach

Table 3 Overall computing speed comparison

Frequency resolution Δf	Feature extraction $t_{\mathrm{fe,WELCH}}$	FS with Filter t_{filter}	FS with Genetic Algorithm t_{GA}	FS with hyperheuristic t_{hyper}
390 Hz	0.049 s	0.03 s with 10 Features	51.54 s with 10 Features	52.08 s with 10 Features
		0.03 s with 50 Features	89.42 s with 50 Features	94.03 s with 50 Features

reduced the feature extraction time, including scaling, to $t_{\mathrm{fe,WELCH}} = 0.036$ s. The subsequent computing time required for a prediction with the pre-trained BR model took $t_{\mathrm{predict}} = 0.0008$ s. The limiting factor for live use was the loading of raw data from the data lake, which took $t_{\mathrm{load}} = 0.137$ s.

Summary and outlook

Within the publication, a modular and scalable data and system architecture was developed that enables the consistent integration of machine-level sensor data, metadata, and machine-learning models across multiple machines and process chains. The architecture supports traceable data access, structured data processing and reproducible model training while allowing seamless transition from offline model development to live application through live calculation and forecasting. By integrating centralized data storage, metadata management and streaming-based communication, the architecture provides the foundation for continuous model monitoring, future drift detection and adaptive retraining, thereby facilitating robust and industrially deployable machine-learning solutions.

Regarding the models, applying a feature extraction method new to the grinding process demonstrated that machine learning models could be utilized much more efficiently. WELCH's spectral density estimation made it possible to analyze trained models in terms of their important frequency components and optimize the sampling rate. The model training demonstrated the importance of combining robust validation strategies with advanced optimization techniques to achieve reliable and economically efficient machine-learning models. Blocked cross-validation revealed that conventional filter-based feature selection tends to overfit and fails to generalize across independent grinding wheels, whereas the hyperheuristic approach consistently delivered superior and more robust performance. Compared to a standalone Genetic Algorithm, the hyperheuristic showed higher predictive accuracy, reduced variance and improved stability, particularly for small populations and high-dimensional feature spaces, underscoring its suitability for complex industrial problems. In contrast, the hyperheuristic required comparable or longer running times than the GA, which was to be expected given its higher-level control and combination of optimization strategies.

Beyond algorithmic optimization, the results emphasize the critical role of signal-processing parameter tuning. The systematic optimization of frequency resolution demonstrated that neither excessively coarse nor overly fine spectral representations are optimal, but that an intermediate resolution maximizes predictive performance while minimizing computational cost. Similarly, optimizing the sampling rate via the maximum frequency revealed that model performance saturates beyond a certain bandwidth, enabling a reduction in data volume without loss of accuracy. Together, the presented architecture highlights a unique pathway to economically optimize machine-learning models by jointly selecting appropriate spectral resolution and sampling rates, while leveraging hyperheuristic optimization to ensure robust generalization in industrially relevant settings.

The presented architecture is currently primarily designed for process monitoring applications, in which model outputs can be computed after completion of a grinding operation. Extending the architecture toward a lambda-based design would enable true real-time capabilities by combining batch processing with low-latency streaming analytics. At present, feature extraction is limited to spectral density estimation based on the WELCH method and is specifically tailored to the subsequent feature selection and modeling stages. However, ongoing advances in hardware performance and the increasing adoption of cloud computing in industrial environments provide a clear pathway for integrating additional time–frequency transformations. The proposed feature selection framework can accommodate extended feature sets, provided that features retain a direct association with physical frequencies. While the application example evaluated four different regression models to illustrate the optimization potential of the methodology, the architecture itself is model-agnostic, allowing straightforward integration of more complex approaches, such as artificial neural networks, in future work.

The proposed methodology for model training is designed to be transferable not only to other grinding processes, such as cylindrical and tool grinding, but also, in principle, to other subtractive manufacturing processes, as it is based on generic frequency characteristics that can likewise be computed across different machining operations. Nevertheless,

a direct transfer of a trained model to a different grinding process is not expected, as process-specific excitation frequencies and machine-dependent damping characteristics differ between grinding operations. Owing to the use of hyperheuristic optimization and robust feature selection, a moderate level of domain shift (e.g., batch fluctuations or similar kinematic conditions) can likely be accommodated with limited retraining effort, whereas larger deviations will require targeted domain adaptation strategies. Adoption strategies for industrial machine learning models could include transfer learning with warm-start retraining, domain adaptation to align feature distributions across processes and incremental learning to handle gradual changes such material variability. In addition, incorporating process metadata could enable hybrid or conditional models that can systematically adapt to different machines or operating conditions. With increasing data availability, more expressive models such as artificial neural networks can be employed to capture complex, non-linear process behavior. In this case, few-shot or even no-shot learning approaches could enable adaptation to new conditions with only minimal or no additional labeled target data.

Funding Open Access funding enabled and organized by Projekt DEAL. This work was supported by the German Research Foundation (DFG) under grant BE 2542/187–1 "Datengetriebene Identifikation und Erklärung von Prozessänderungen infolge von werkstück- und werkzeugseitigen Chargenschwankungen beim Flachschleifen " (project number 552756260).

Declarations

Conflict of interest The authors have no relevant financial or nonfinancial interests to disclose.

References

Allen, J. B., & Rabiner, L. R. (1977). A unified approach to short-time Fourier analysis and synthesis. *Proceedings of the IEEE, 65*(11), 1558–1564. https://doi.org/10.1109/PROC.1977.10770

Barandas, M., Folgado, D., Fernandes, L., Santos, S., Abreu, M., Bota, P., Liu, H., Schultz, T., & Gamboa, H. (2020). TSFEL: Time series feature extraction library. *SoftwareX, 11*, Article 100456. https://doi.org/10.1016/j.softx.2020.100456

Beckers, A., Hommen, T., Becker, M., Kornely, M. J., Reuter, E., Grünert, G., Ortjohann, L., Jacob, J., Niemietz, P., Barth, S., & Bergs, T. (2022). Digitalized manufacturing process sequences – Foundations and analysis of the economic and ecological potential. *CIRP Journal of Manufacturing Science and Technology, 39*, 387–400. https://doi.org/10.1016/j.cirpj.2022.09.001

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization. *ACM Computing Surveys, 35*(3), 268–308. https://doi.org/10.1145/937503.937505

Brinksmeier, E., Aurich, J. C., Govekar, E., Heinzel, C., Hoffmeister, H.-W., Klocke, F., Peters, J., Rentsch, R., Stephenson, D. J., Uhlmann, E., Weinert, K., & Wittmann, M. (2006). Advances in modeling and simulation of grinding processes. *CIRP Annals, 55*(2), 667–696. https://doi.org/10.1016/j.cirp.2006.10.003

Burggräf, P., Steinberg, F., Sauer, C. R., & Nettesheim, P. (2024). Machine learning implementation in small and medium-sized enterprises: Insights and recommendations from a quantitative study. *Production Engineering, 18*, 751–764. https://doi.org/10.1007/s11740-024-01274-2

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society, 64*(12), 1695–1724. https://doi.org/10.1057/jors.2013.71

Christ, M., Braun, N., Neuffer, J., & Kempa-Liehr, A. W. (2018). Time Series FeatuRe Extraction on basis of scalable hypothesis tests (tsfresh – A Python package). *Neurocomputing, 307*, 72–77. https://doi.org/10.1016/j.neucom.2018.03.067

Daolio, F., Liefooghe, A., Verel, S., Aguirre, H., & Tanaka, K. (2017). Problem features versus algorithm performance on rugged multiobjective combinatorial fitness landscapes. *Evolutionary Computation, 25*, 555–585. https://doi.org/10.1162/evco_a_00193

Deloitte Touche Tohmatsu Limited (2025). Navigating challenges to implementation. *Smart Manufacturing and Operations Survey*. Retrieved 03.06.2025, from https://www2.deloitte.com/us/en/insights/industry/manufacturing/2025-smart-manufacturing-survey.html

Elgamel, S. (2012). Empirical Mode Decomposition Complexity. *The International Conference on Electrical Engineering*, 8, S. 1–9. https://doi.org/10.21608/iceeng.2012.30659

Esling, P., & Agon, C. (2012). Time-series data mining. *ACM Computing Surveys, 45*(1), 1–34. https://doi.org/10.1145/2379776.2379788

Fulcher, B. D. (2017) *Feature-based time-series analysis,* https://doi.org/10.48550/arXiv.1709.08055

Furner, J. (2020). Definitions of "metadata": A brief survey of international standards. *Journal of the Association for Information Science and Technology, 71*(6), 33–42. https://doi.org/10.1002/asi.24295

Ganser, P., Venek, T., Rudel, V., & Bergs, T. (2021). DPART – A digital twin framework for the machining domain. *MM Science Journal*. https://doi.org/10.17973/MMSJ.2021_11_2021168

Godoy Neto, R. F.; Marchi, M.; Martins, C.; Aguiar, P. R. P.; Bianchi, E. (2014). Monitoring of Grinding Burn by AE and Vibration Signals. *Proceedings of the 6th International Conference on Agents and Artificial Intelligence: SCITEPRESS - Science and and Technology Publications*, pp. 272–279. https://doi.org/10.5220/0004753602720279

Hastie, T.; Tibshirani, R.; Friedman, J. H. (2017). *The elements of statistical learning. Data mining, inference, and prediction* (2nd ed.) Springer Series in Statistics. New York, NY: Springer. ISBN: 0387848576.

Huang, Z., Shen, Y., Li, J., Fey, M., & Brecher, C. (2021). A survey on AI-driven digital twins in Industry 4.0: Smart manufacturing and

advanced robotics. *Sensors, 21*(19), Article 6340. https://doi.org/10.3390/s21196340

(ISO 10007, 2017)International Organization for Standardization. (2021). Quality management — Guidelines for configuration management (ISO 10007:2017).

James, G.; Witten, D.; Hastie, T.; Tibshirani, R. (2021). *An Introduction to Statistical Learning*. New York, NY: Springer US. ISBN: 978–1–0716–1417–4.

Jensen, S. K., Pedersen, T. B., & Thomsen, C. (2017). Time series management systems: A survey. *IEEE Transactions on Knowledge and Data Engineering, 29*(11), 2581–2600. https://doi.org/10.1109/TKDE.2017.2740932

Jovic, A.; Brkic, K.; Bogunovic, N. (2015). A review of feature selection methods with applications. *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO): IEEE*, pp. 1200–1205. https://doi.org/10.1109/MIPRO.2015.7160458

Khouas, A. R.; Bouadjenek, M. R.; Hacid, H.; Aryal, S. (2024). *Training Machine Learning models at the Edge: A Survey,* https://doi.org/10.48550/arXiv.2403.02619

Krajnik, P., Wegener, K., Bergs, T., & Shih, A. J. (2024). Advances in modeling of fixed-abrasive processes. *CIRP Annals, 73*(2), 589–614. https://doi.org/10.1016/j.cirp.2024.05.001

Lambora, A.; Gupta, K.; Chopra, K. (2019). Genetic Algorithm- A Literature Review. *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon): IEEE*, pp. 380–384. https://doi.org/10.1109/COMITCon.2019.8862255

Landau, H. J. (1967). Sampling, data transmission, and the Nyquist rate. *Proceedings of the IEEE, 55*(10), 1701–1706. https://doi.org/10.1109/PROC.1967.5962

Lei, L., & Kun, S. (2017). Speaker recognition using wavelet packet entropy, I-Vector, and cosine distance scoring. *Journal of Electrical and Computer Engineering, 2017*, 1–9. https://doi.org/10.1155/2017/1735698

Lundberg, S.; Lee, S.-I. (2017) *A Unified Approach to Interpreting Model Predictions,* https://doi.org/10.48550/arXiv.1705.07874

Mahata, S., Shakya, P., & Babu, N. R. (2021). A robust condition monitoring methodology for grinding wheel wear identification using Hilbert Huang transform. *Precision Engineering, 70*, 77–91. https://doi.org/10.1016/j.precisioneng.2021.01.009

Mathis, C. (2017). Data lakes. *Datenbank-Spektrum, 17*(3), 289–293. https://doi.org/10.1007/s13222-017-0272-7

Mouli, D. S. B.; Rameshkumar, K. (2020). Acoustic Emission-Based Grinding Wheel Condition Monitoring Using Decision Tree Machine Learning Classifiers. *Advances in Materials and Manufacturing Engineering*. Lecture Notes in Mechanical Engineering, pp. 353–359. https://doi.org/10.1007/978-981-15-1307-7_39

Oppenheim, A. V.; Buck, J. R.; Schafer, R. W. (1999). *Discrete-time signal processing*. (2nd ed). Prentice Hall signal processing series. Upper Saddle River, NJ: Prentice-Hall. ISBN: 978–0137549207.

Pandiyan, V., Shevchik, S., Wasmer, K., Castagne, S., & Tjahjowidodo, T. (2020). Modelling and monitoring of abrasive finishing processes using artificial intelligence techniques: A review. *Journal of Manufacturing Processes, 57*, 114–135. https://doi.org/10.1016/j.jmapro.2020.06.013

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Müller, A.; Nothman, J.; Louppe, G.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research,* https://doi.org/10.48550/arXiv.1201.0490

Rebala, G.; Ravi, A.; Churiwala, S. (2019). *An introduction to machine learning*. Springer Cham. https://doi.org/10.1007/978-3-030-15729-6

Krishnan, P. S., & Rameshkumar, K. (2021). Grinding wheel condition prediction with discrete hidden Markov model using acoustic emission signature. *Materials Today: Proceedings, 46*, 9168–9175. https://doi.org/10.1016/j.matpr.2019.12.428

Sauter, E., Sarikaya, E., Winter, M., & Wegener, K. (2021). In-process detection of grinding burn using machine learning. *The International Journal of Advanced Manufacturing Technology, 115*, 2281–2297. https://doi.org/10.1007/s00170-021-06896-9

Sawadogo, P., & Darmont, J. (2021). On data lake architectures and metadata management. *Journal of Intelligent Information Systems, 56*(1), 97–120. https://doi.org/10.48550/arXiv.2107.11152

Sörensen, K. (2015). Metaheuristics—the metaphor exposed. *International Transactions in Operational Research, 22*(1), 3–18. https://doi.org/10.1111/itor.12001

Storcheus, D.; Rostamizadeh, A.; Kumar, S. (2015). A Survey of Modern Questions and Challenges in Feature Extraction. *Proceedings of the 1st International Workshop on Feature Extraction: Modern Questions and Challenges at NIPS 2015*, pp. 1–18.

(Theodoridis, 2009)Book Theodoridis, S. (2009). *Pattern recognition*. (4th ed). Burlington, MA: Academic Press. ISBN: 9780080949123

Torrence, C., & Compo, G. P. (1998). A practical guide to wavelet analysis. *Bulletin of the American Meteorological Society, 79*(1), 61–78.

Vashishtha, G., Chauhan, S., Singh, M., & Kumar, R. (2021). Bearing defect identification by swarm decomposition considering permutation entropy measure and opposition-based slime mould algorithm. *Measurement, 178*, Article 109389. https://doi.org/10.1016/j.measurement.2021.109389

Vetterli, M.; Kovačević, J.; Goyal, V. K. (2014). *Foundations of Signal Processing*: Cambridge University Press. ISBN: 9781139916578.

Vetterli, M.; Kovačević, J. (1995). *Wavelets and subband coding*. Prentice Hall signal processing series. Englewood Cliffs, NJ: Prentice Hall PTR. ISBN: 0–13–097080–8.

Wang, D., Tan, D., & Liu, L. (2018). Particle swarm optimization algorithm: An overview. *Soft Computing, 22*(2), 387–408. https://doi.org/10.1007/s00500-016-2474-6

Wang, Y., & He, P. (2023). Comparisons between fast algorithms for the continuous wavelet transform and applications in cosmology: The 1D case. *RAS Techniques and Instruments, 2*(1), 207–323. https://doi.org/10.1093/rasti/rzad020

Warren Liao, T. (2010). Feature extraction and selection from acoustic emission signals with an application in grinding wheel condition monitoring. *Engineering Applications of Artificial Intelligence, 23*(1), 74–84. https://doi.org/10.1016/j.engappai.2009.09.004

Welch, P. (1967). The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics, 15*(2), 70–73. https://doi.org/10.1109/TAU.1967.1161901

Yang, Z., & Yu, Z. (2011). Grinding wheel wear monitoring based on wavelet analysis and support vector machine. *The International Journal of Advanced Manufacturing Technology, 62*, 107–121. https://doi.org/10.1007/s00170-011-3797-1

Yang, Z., & Yu, Z. (2013). Experimental study of burn classification and prediction using indirect method in surface grinding of AISI 1045 steel. *The International Journal of Advanced Manufacturing Technology, 68*, 2439–2449. https://doi.org/10.1007/s00170-013-4882-4