UDAVA: An Unsupervised Learning Pipeline for Sensor Data Validation in Manufacturing

Erik Johannes Husom
SINTEF Digital, Norway
erik.husom@sintef.no

Simeon Tverdal
SINTEF Digital, Norway
simeon.tverdal@sintef.no

Arda Goknil
SINTEF Digital, Norway
arda.goknil@sintef.no

Sagar Sen
SINTEF Digital, Norway
sagar.sen@sintef.no

ABSTRACT

Manufacturing has enabled the mechanized mass production of the same (or similar) products by replacing craftsmen with assembly lines of machines. The quality of each product in an assembly line greatly hinges on continual observation and error compensation during machining using sensors that measure quantities such as position and torque of a cutting tool and vibrations due to possible imperfections in the cutting tool and raw material. Patterns observed in sensor data from a (near-)optimal production cycle should ideally recur in subsequent production cycles with minimal deviation. Manually labeling and comparing such patterns is an insurmountable task due to the massive amount of streaming data that can be generated from a production process. We present UDAVA, an unsupervised machine learning pipeline that automatically discovers process behavior patterns in sensor data for a reference production cycle. UDAVA performs clustering of reduced dimensionality summary statistics of raw sensor data to enable high-speed clustering of dense time-series data. It deploys the *model as a service* to verify batch data from subsequent production cycles to detect recurring behavior patterns and quantify deviation from the reference behavior. We have evaluated UDAVA from an AI Engineering perspective using two industrial case studies.

ACM Reference Format:
Erik Johannes Husom, Simeon Tverdal, Arda Goknil, and Sagar Sen. 2022. UDAVA: An Unsupervised Learning Pipeline for Sensor Data Validation in Manufacturing. In *1st Conference on AI Engineering - Software Engineering for AI (CAIN'22), May 16–24, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3522664.3528603

1 INTRODUCTION

Repetitive manufacturing is a form of mass production that relies on making large numbers of identical goods/parts in a continuous flow. It is used by manufacturers committed to a specific production rate. For instance, Renault produces 1500 cylinder heads per day for electric car engines in one of its production lines in the bodywork assembly plant [35] in Valladolid, Spain. These cylinder heads are repetitively manufactured using a fleet of multi-axis CNC (computer numerical control) milling machines chipping raw material at high speed. ITP Aero [22], a Spanish turbine manufacturer, delivers turbine discs with *fir tree slots* where each slot must be identical and cut using the *broaching* process [13]. Data acquisition systems on edge devices (e.g., program logic controllers) acquire sensor data at sub-microsecond sampling frequencies during repetitive manufacturing for real-time decision making and post mortem analysis in case of product defects or production failures.

Sensor data acquired during machining can reveal transitions in process behavior reflecting normal operation or process shifts and drifts [16] leading to product defects. Process shifts and drifts are unexplained or unexpected trends of a measured process parameter(s) away from its intended target value in time-ordered analysis. They affect the ability to produce goods within specifications. One common source of process shifts is the initial, manual setup of the manufacturing line (e.g., sensor calibration [29]), which has to take place each time a new lot (i.e., a considerable quantity of goods/parts) is produced. A process drift goes toward shifting in one direction over time. The typical sources of process drifts are sensor faults [25], tool wear [42], workpiece surface quality [23], and chip evacuation mechanisms [8]. It is hard to detect a process drift as it may get hidden behind others.

High-volume and velocity multivariate sensor data acquired during manufacturing introduce a challenging task for human operators to find diverse patterns of interest and track their deviations (e.g., process shifts and drifts) over multiple production cycles. Therefore, we investigate in this paper whether we can engineer an AI system to (a) automatically discover *reference patterns* representing modes of process behavior in manufacturing data from a reference production cycle and (b) validate data in subsequent production cycles by identifying the recurrence of these patterns.

In this paper, we present an unsupervised machine learning pipeline for sensor data validation in manufacturing, i.e., UDAVA automatically discovering process behavior patterns in sensor data. Machine Learning (ML) pipelines for data validation in the literature cluster raw time series data obtained directly from manufacturing sensors. They use the raw data similarity as a distance metric to divide time series data into sub sequences [2]. The most common distance metric is the Euclidean distance [14] which is limited to fixed sizes of time series and sensitive to noise and distortion. Dynamic time warping (DTW) [31] overcomes some of the limitations of Euclidean distance but has quadratic computational complexity

CAIN'22, May 16–24, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9275-4/22/05…$15.00
https://doi.org/10.1145/3522664.3528603

and hence does not scale very well with large data sets. Given the large data sets of historical data in manufacturing, UDAVA extracts features from the sub sequences of time series data and clusters these features instead of raw data.

UDAVA takes, as input, time-varying sensor data from one or more reference production cycles during repetitive manufacturing. A reference production cycle is a (near-)optimal production cycle where the manufactured products are within acceptable tolerances of the specification, with almost no defects, preferably low energy consumption, and minimal waste generation. UDAVA performs clustering on *vectors of summary statistics* (i.e., feature vectors) derived from raw sensor data over configurable window sizes. The raw sensor data is automatically labeled based on clusters representing distinct behavior modes. UDAVA wraps, as a *web service*, the AI model represented by clusters/behavior modes discovered in the reference sensor data. The web service can be updated with a new model when the product or process parameters change. It is containerized as a Docker container and deployed on edge devices or cloud infrastructures where new data from subsequent production cycles is acquired continually.

UDAVA essentially solves an unsupervised learning problem in the manufacturing domain. However, numerous challenges arise in the engineering and deployment of UDAVA in industrial production environments. Therefore, we have analyzed and evaluated UDAVA from data science and AI engineering perspectives (see our research questions in Section 6). We have used reference data sets from two industrial case studies: (a) broaching fir tree slots for jet engine turbine discs performed at CFAA - Advanced Manufacturing Centre for Aeronautics, Spain, while data acquisition occurs on an edge device developed by SAVVY data systems [36], and (b) high-speed CNC milling of car engine cylinder heads at Renault's Valladolid factory, Spain, where data is acquired in an edge device CASIP and high-frequency data persisted in the cloud infrastructure KASEM E-maintenance both developed by Predict [34] in France.

The paper is structured as follows. Sections 2-3 present the background and related work. In Section 4, we describe the technical details of UDAVA. Section 5 presents the deployment of UDAVA. Section 6 reports on the evaluation. Section 7 concludes the paper.

2 BACKGROUND

UDAVA is engineered as a data pipeline (Section 2.1) and employs unsupervised learning (Section 2.2). We evaluated UDAVA from an AI engineering perspective (Section 2.3).

2.1 Data Pipelines with Data Version Control

A data pipeline is a digital infrastructure facilitating data processing. ML applications generally require complex data pipelines that can handle all steps from processing raw data to producing a trained (and possibly deployed) ML model [3]. The first part of such pipelines involves data cleaning, feature engineering, and data restructuring for the input format of the ML algorithm. It is followed by building and training the model. Developing an ML model involves running various experiments to fine-tune configuration and control parameters. Therefore, there is a need to track source code, input and output data, control parameters, and models.

Traditional version control systems like *Git* [27] easily keep track of changes in source code and control parameters but are ill-suited for tracking big data and binary files, e.g., ML models. We use the Data Version Control (DVC) framework [21, 24] to support big files in our pipeline. A DVC pipeline is created by defining a set of stages. Each stage has a run command(s) concerning its dependencies to other stages. The dependencies usually involve input data, control parameters, expected output, and source code. DVC can automatically detect and track changes in any element of the pipeline. The cache stores output of data processing stages. DVC computes output hashes to compare the current output with the previous ones of any given stage. The advantage is that DVC can skip the execution of certain stages and instead fetch the correct output from the cache if they have already run in the same configuration. The execution time is significantly reduced for large numbers of experiments. We can configure DVC to track performance metrics for any model created using the pipeline. And, we can analyze the history of all experiments with perfect reproducibility.

2.2 Unsupervised learning

Unsupervised learning refers to machine learning algorithms identifying patterns in data sets without any labels or human guidance. One unsupervised learning method is clustering observations in a data set based on their characteristics. It aims to find a cluster configuration with the maximum similarity between in-cluster observations and the maximum dissimilarity between different clusters. Measuring the Euclidean distance between observations gives observation similarity. UDAVA employs two clustering algorithms, i.e., K-means and the mean shift algorithm.

K-means clustering [28, 43] is a centroid-based cluster algorithm that defines a cluster with a vector representing the cluster center (centroid). It requires a predefined number of clusters. First, each observation in the data set is randomly assigned to a cluster. Then, the algorithm computes the centroids using the random assignments. It redistributes the observations to new clusters based on the closest centroids. The centroids are recomputed until we have no changes on the cluster assignments or reach a predefined maximum number of iterations. The cluster centroids are identified by using a cluster label, i.e., an integer from 1 to N, where N is the total number of clusters. We can improve the efficiency of the K-means by using only a subset, a mini-batch of randomly sampled data from the data set in each iteration during model training [37]. Mini-batches significantly reduce the time needed for the model to converge and make the K-means suitable to deal with large amounts of data.

The mean shift [10, 18] is a density-based algorithm searching dense areas within the observation space to identify clusters. It automatically decides the number of clusters, while the K-means algorithm needs a fixed number. However, the mean shift is much more computationally expensive to run.

2.3 AI Engineering

Artificial intelligence (AI) and machine learning (ML) have been increasingly adopted by the industry. It has been observed over a dozen case studies that deploying industry-strength AI systems (i.e., systems that include AI components) and ML models in those

systems proves to be challenging [7]. The problems most companies experience in AI systems are concerned with a range of topics, including data quality, design methods and processes, the performance of ML models, and deployment and compliance.

Bosch et al. [7] define the term "AI engineering" as an extension of software engineering with new processes and technologies for the development and evolution of AI systems. The goal of AI engineering is to address the engineering challenges of AI systems from the software engineering point of view. To this end, Bosch et al. propose a research agenda that provides (i) typical evolution patterns concerned with AI adoption that companies experience, (ii) an overview of the engineering challenges surrounding ML solutions, and (iii) open items that need to be addressed.

Below we briefly describe some AI engineering dimensions we consider while devising and evaluating UDAVA.

- **Data versioning and dependency management**. The quality of data used for training is crucial to achieving the high performance of ML models. Since data pipelines tend to be less robust than software pipelines, it is important to provide data quality management solutions. Data quality can be supported by simple checks for data being in range or present. More advanced checks ensure that the average for a data window stays constant or the statistical distribution of data remains similar. Ensuring data quality can be particularly challenging for different types of data in use.
- **Deployment infrastructure**. Independent from centralized or distributed ML approaches, models need to be deployed in systems in the field. Companies need a deployment infrastructure that reliably deploys subsequent model versions, measures model performance, and raises warnings for anomalous behavior. Deployment of ML models may require a substantial change in the system architecture.
- **Quality attributes**. The key challenge of data science for ML models is to achieve high accuracy and precision. On the other hand, several other quality attributes, including computation performance, the number of inferences per time unit, real-time properties, and system robustness, are relevant from the AI engineering perspective. It is a challenge to design and implement an AI system that meets the quality requirements of the ML components in the system.
- **Integration of models and components**. Since an AI system has not only AI components, companies need to integrate ML models with the traditional software components of the system. Software verification techniques need to be adapted to check whether AI and regular software components are integrated seamlessly. Depending on the criticality of ML models, the validation and verification activities need to be more elaborate and strict.

3 RELATED WORK

Approaches to time series clustering, in general, have been reviewed by Aghabozorgi et. al. [1], and Alqahtani et. al. [2] provide a review of (deep)-clustering approaches to time series data. Dogan et. al. [15] present a comprehensive overview of ML methods for manufacturing. According to these surveys, there is a considerable focus on using clustering to detect patterns in the end product (e.g., [26]). Clustering time-varying sensor data from the manufacturing process has been used to address *anomaly detection* and *predictive maintenance* (e.g., [4]). Predictive maintenance in manufacturing often assumes that sensor data is high quality without sensor faults [25]. We believe that, like in UDAVA, unsupervised learning should steer its focus towards sensor data validation.

Euclidean distance is used as a deviation metric in clustering to compute the distance between two time series data. One limitation of Euclidean distance is that distance only between fixed-length time series data can be computed. This metric is very sensitive to noise and distortion and less robust to non-linearity. Dynamic Time Warping (DTW) [31] overcomes the limitation of Euclidean distance by aligning (warping) the series before computing the distance. However, two temporal points with different local structures might be mistakenly matched. This issue is addressed by the shape Dynamic Time Warping (shapeDTW) [51] using point-wise local structural information. DTW has been used for manufacturing applications (e.g., bearing fault diagnosis [44] and predictive maintenance [40]). It has quadratic computational complexity and does not scale with large data sets in manufacturing.

Clustering large data sets requires dimensionality reduction [45]. Widely used methods for dimensionality reduction include Principal Component Analysis (PCA) [41] extracting features mutually uncorrelated, K-grams [49] preserving the sequential nature of series, Discrete Fourier Transforms [17] extracting features from time-frequency characteristics, and Shapelets [48]. Like some of these methods, UDAVA employs feature extraction, a form of dimension reduction, to lower the cost of high-dimensional data and achieve higher clustering accuracy.

Engineering AI systems [7] (e.g., infusing unsupervised learning into manufacturing environments) has not been discussed extensively. Angelopoulos et. al. [5] present learning algorithms for Industry 4.0 with little focus on how to design and deploy them. Some works [9, 46, 47] discuss unsupervised learning for predictive maintenance and anomaly detection without mentioning AI engineering aspects. Our paper presents the field knowledge of how ML models are deployed, maintained, tested, from which researchers can benefit.

4 UDAVA APPROACH

The process in Figure 1 presents an overview of our approach. The main steps of the approach are as follows.

(1) **Preprocessing reference data**. UDAVA splits time series data into subsequences and extracts statistical features from them. Output feature vectors represent the subsequences.
(2) **Unsupervised learning of clusters.** UDAVA performs cluster analysis on the feature vectors. It assigns each feature vector to a cluster/category. The cluster analysis produces a model consisting of several cluster centers; each cluster center defines a process behavior pattern.
(3) **Labeling and validating new data.** UDAVA computes the feature vectors of the time series data to be validated. It calculates a *deviation metric* by checking the sum of distances between feature vectors and cluster centers.

UDAVA preprocesses both reference (training) and production time series data (data to be validated). The feature vectors extracted

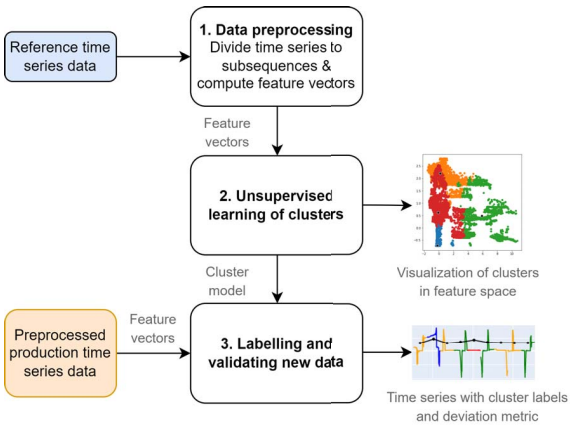

Figure 1: Overview of UDAVA.

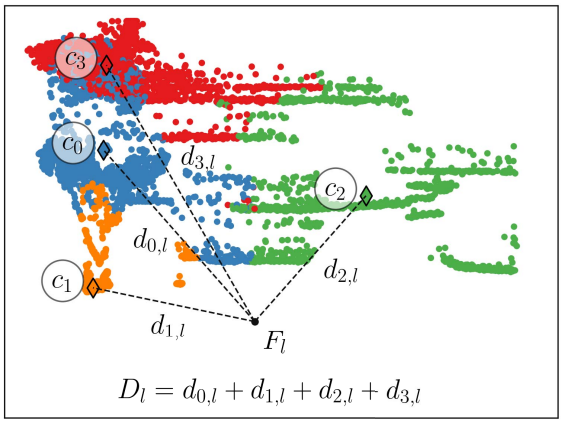

Figure 2: Example clusters detected by UDAVA in 2D feature space. Each colored dot represents one observation (feature vector) of the data set; the different colors (red, orange, blue, and green) correspond to four clusters in the model. Diamond-shaped markers indicate the cluster centers. The annotations illustrate the calculation of the deviation metric D_l for a feature vector F_l, shown as a black dot.

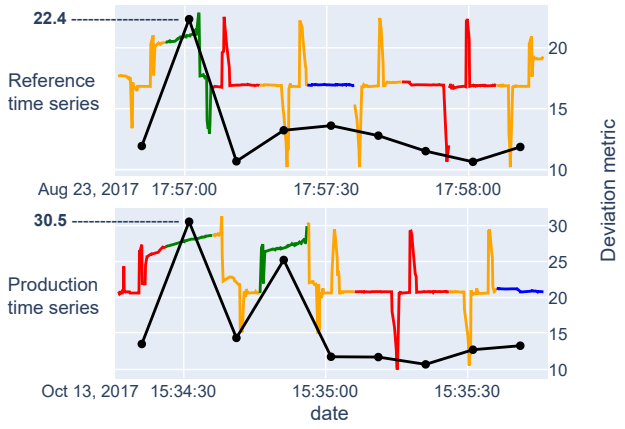

Figure 3: Raw data from an example data set (multicolored line) with the deviation metric D (black line), for the reference (top) and production (bottom) data. The subsequences are colored based on the cluster they belong to. The maximum value of the deviation metric is indicated in both plots.

from the production time series data are input to Step 3 in Figure 1. The following subsections explain each step in Figure 1.

4.1 Reference Data Preprocessing

Information in raw time series exists as relations between consecutive time-varying data points. Clustering high volumes of raw time series is computationally expensive. UDAVA extracts features from data subsequences, thus removing temporal dimension. If the finer granularity is needed, it uses the sliding window technique [11, 17] requiring more computing time. Feature extraction is done on each subsequence, resulting in a feature vector (as we call it).

Summary statistics of subsequences can be used as features to generate a feature vector. UDAVA employs six basic features that convey essential information of sub-sequences: *mean*, *median*, *standard deviation*, *variance*, *range*, and *frequency*. The range is the difference between the minimum and maximum values of the subsequence, and the frequency is calculated using Fast Fourier Transform (FFT). Features are given equal weight in clustering and are scaled based on the following equation:

$$f = \frac{f' - \mu}{\sigma}, \tag{1}$$

where μ and σ are the mean and standard deviation of the feature f' before scaling and f is the scaled feature.

4.2 Unsupervised learning of clusters

UDAVA runs a clustering algorithm to automatically assign each feature vector into clusters (Step 2 in Figure 1). The output of the algorithm is a cluster model consisting of cluster centers for reference data. UDAVA is configurable for a wide range of clustering algorithms available in the Python package Scikit-learn [32]. For instance, the standard K-means algorithm can be used for its efficiency when the number of clusters is specified. UDAVA can be configured with the mean shift algorithm if one prefers the number of clusters to be automatically decided. In K-means clustering, the similarities between observations in the data set are evaluated by calculating the multidimensional Euclidean distance based on all the features in each feature vector. Figure 2 illustrates example clusters using only two features in the feature vectors.

4.3 Labeling and validating new data

UDAVA uses the cluster model to label and validate new data obtained from production (Step 3 in Figure 1). Production data is preprocessed by UDAVA in the same way as reference data (see Subsection 4.1). The cluster model is employed to assign cluster labels to each feature vector of the production data. Figure 3 presents some example cluster labels. The multicolored line represents sensor data. Its color changes according to the cluster. The cluster labels are plotted back into temporal space by assigning the feature vectors' labels to the corresponding subsequence.

UDAVA detects deviating process behavior to validate production data. It computes the distance of the feature vectors obtained from production data with the cluster centers in the cluster model obtained from the reference data. The *deviation metric* is computed for each feature vector of the production time series data. The reference data is collected during (near-)optimal production cycles to provide cluster centers that can serve as a baseline for the comparison with the production data. The deviation metric is about how close a new observation (a feature vector) is to the existing cluster centers defined by the model. Cluster centers in the model are defined by the mean of the features of all feature vectors in each cluster. Let the total number of clusters be N, and the index of each cluster c be $i \in 1, ..., N$. The cluster center c_i is

$$c_i = [\mu_{i,1}, \mu_{i,2}, .., \mu_{i,j}, ..., \mu_{i,M-1}, \mu_{i,M}], \quad (2)$$

where $\mu_{i,j}$ is the mean of feature j of all the feature vectors in cluster i. The index j runs from 1 to M, where M is the number of features in each feature vector. To validate production data, we first compute the Euclidean distance in the multidimensional feature space $d_{i,l}$ each feature vector F_l has to each cluster center c_i, where $l \in 1, \ldots, L$ is the feature vector's index in the data set.

$$d_{i,l} = \sqrt{(f_1 - \mu_{i,1})^2 + \cdots + (f_j - \mu_{i,2})^2 + \cdots + (f_M - \mu_{i,M})^2)}, \quad (3)$$

where f_j represents the features for the given feature vector F_l. These distances are then summed using the following formula:

$$D_l = \sum_{i=1}^{N} d_{i,l}. \quad (4)$$

D is the deviation metric and can be monitored over time to check how much the production data deviates from the reference data. Figure 2 illustrates how D_l, marked by the black dot, is calculated for feature vector F_l.

We consider the sum of distances as a measure of deviation because it gives how far an observation is from the known clusters. Feature vector F_l of cluster i might be far from cluster center c_i, relative to the other data points in the cluster (i.e., $d_{i,l}$ is large). It does not necessarily mean it is a deviation since it might be close to the boundary against another cluster. Please check the red data points in the center of Figure 2. Even though they are relatively far from the red cluster center, they are close enough and may share attributes with the data points in the green and blue clusters. Therefore, the distance of a feature vector to any individual cluster center may fail as a metric to discover deviations. However, the sum of distances increases when observations are far from *all* the cluster centers, e.g., observations in the periphery and outside the plotted area in Figure 2. A feature vector always has a non-zero distance to at least one of the cluster centers. Therefore, the deviation metric is never zero for any feature vector in the reference and production data sets. We need to interpret the value of the deviation metric of a given observation with other values from the same production data set or the reference data set. A spike in the deviation metric or a higher average for a production cycle compared to the reference data set are signs of deviations or significant changes in the production data.

The black dotted line in Figure 3 represents deviation metric D. Each dot shows the value of D_l for feature vector F_l that corresponds to the subsequence of the sensor data for that time slot. The two plots show similar production cycles for the reference time series (top) and the production time series (bottom). There is a significant time gap between the two time series. The former was recorded in August 2017, and the latter was in October 2017. The deviation metric in both plots shows a spike where the green cluster appears. It indicates that the sensor exhibits a deviating behavior known as *sensor drift*. The spike in the reference data reaches a value of $D = 22.4$, while the highest one in the production data reaches $D = 30.5$. The increase of the deviation metric can also be used to detect the occurrence of faults in the manufacturing process. Furthermore, we can use cluster labels to discover repeating patterns. If one cluster is observed for faulty behavior, we can flag each occurrence of the given cluster label to identify similar behaviors.

5 DEPLOYMENT OF UDAVA

We can deploy UDAVA on a standalone machine, edge device, or the cloud as a docker container. The pipeline requires access to a time series database, such as InfluxDB, or an API provided by a data acquisition system used by manufacturers. The ML models as a service are integrated into UDAVA as a web server or a docker container invoked by data owners. The service is deployed on-premises on an edge device or the cloud of data owners. Therefore, the model as a service has a simple and open API (without security authentication) for receiving data and sending clusters and deviations as output. Figure 4 presents the structure of the API. The service back-end may contain multiple models, and each model is assigned a unique identifier (UID). UIDs are retrievable from the UI of the UDAVA service installed on edge or by querying the API using a REST GET call. The API has a POST method receiving data for validation. A client, e.g., a company building the edge device to acquire data, sends a JSON request that contains the UID of the model to be used, in addition to the input time series data and its variable names. Listing 1 presents an example JSON request.

The request starts with the model identifier (Line 1 in Listing 1). The names of the data columns follow the identifier (Line 3). The first column contains timestamps for each data point in the second column (Lines 5-10). The example input data contains timestamps and spindle torque values. The spindle torque is sampled every five seconds, and the input data has six data points.

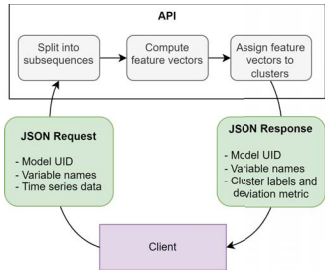

Figure 4: The API Overview.

The back-end of the API splits the input data into subsequences, and feature vectors are computed for each subsequence (see Figure 4). Each feature vector is compared to the cluster centers in the cluster model and then assigned to the cluster to which it is closest. UDAVA's service returns the deviation metric for each feature vector and the cluster to which the vector belongs in the given

```
{"param": {"modeluid": "618b9b95-7805"},
 "scalar": {
    "headers": ["date","SpindleTorque"],
    "data": [
      ["2017-08-23 17:57:00", 101.2],
      ["2017-08-23 17:57:05", 101.3],
      ["2017-08-23 17:57:10", 101.2],
      ["2017-08-23 17:57:15", 101.3],
      ["2017-08-23 17:57:20", 101.4],
      ["2017-08-23 17:57:25", 101.5], ] } }
```

Listing 1: Example JSON request to UDAVA API.

```
{"param": {"modeluid": "618b9b95-7805"},
 "scalar": {
    "headers": ["date","cluster","metric"],
    "data": [
      ["2017-08-23 17:57:05", 0, 16.77],
      ["2017-08-23 17:57:20", 1, 38.74], ] } }
```

Listing 2: Example JSON response from UDAVA API.

cluster model. They are returned as a time series with timestamps. Listing 2 presents the JSON response for the request in Listing 1.

The model used in the example splits the input sequence into subsequences of three values without overlap. Hence, given input data with six data points, we obtain two input subsequences, and the output has two data points (Lines 5-6 in Listing 2). The timestamps in the first column in the output JSON correspond to the middle of each original subsequence. The second column in the data points (header `cluster`) is the cluster label assigned by the model; the third column (header `metric`) is the deviation metric D.

The schema for JSON requests and responses is extensible. We can extend the headers in the request to process data from multiple sensors. Furthermore, we can introduce other metrics (in addition to deviation metric D) to provide additional information on sensor data validation. We containerize the API and cluster model as a Docker container to support easy deployment. The API can be used in the cloud infrastructures of data owners (manufacturing companies). Therefore, it can send and receive data without encryption and implement security measures such as authentication.

UDAVA builds the model as often as necessary, but mostly when a new reference production cycle (minimal tool wear and high product quality) is available and when start and end timestamps are available after production. When data freshness is low, UDAVA employs a new reference cycle to create a new model. UDAVA stores and versions models as binary files using a chronological system in Git (with DVC).

The models (clusters obtained from reference data) can infer in N/f second (N is the length of the subsequence, and f is the sampling frequency) with a delay due to communication with the service (running as a Docker container on edge). UDAVA invokes the model as a service with, for instance, six values (Listing 1), and the model finds two clusters for N=3 (Listing 2). The inference occurs in 0.9 secs (about 1Hz). The output is recommendations to operators to stop machining if necessary. However, one may also automatically pause machining based on faults (sensor drift).

With UDAVA, we emphasize the engineering need to handle evolving data obtained at high velocities using a pipeline that can synthesize services to detect behavior and deviations in production. The traditional unsupervised learning flow expects a curated data set where data does not often evolve with time.

6 EVALUATION

In this section, we investigate, based on two industrial data sets, the following Research Questions (RQ)s:

- ***RQ1.*** *Can UDAVA discover process behavior patterns in sensor data for a reference production cycle?*
- ***RQ2.*** *Does comparing process behavior patterns reveal process shifts and drifts in subsequent production cycles?*
- ***RQ3.*** *How can UDAVA be deployed, tested and maintained in industrial production environments?*

6.1 Subjects of the Evaluation

We applied UDAVA to discover sensor data anomalies in two industrial case studies: (i) broaching of turbine discs for airplane jet engines and (ii) milling of cylinder heads for car engines.

6.1.1 Aerospace case study. This case study involves the data set of the broaching of turbine discs for jet engines. Broaching is a manufacturing process for forming internal or external round, flat, or contoured surfaces. A broaching machine pushes a multi-toothed cutting tool, called a broach, into a workpiece to remove material. Slots of various dimensions are cut at high production rates. Our broaching machine data set was collected from three broaching tools in a broach tool holder for around three hours on the 24th of May 2021 (see Figures 5 and 6).

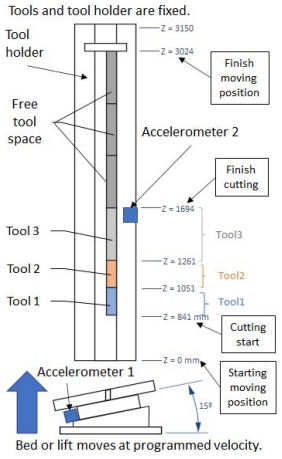

Figure 5: Broaching machine.

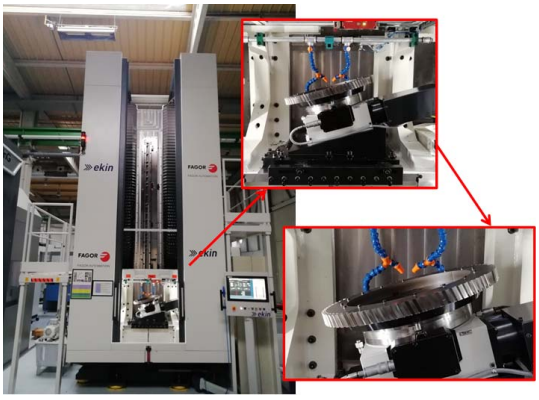

Figure 6: Broaching machine setup and fir tree slots.

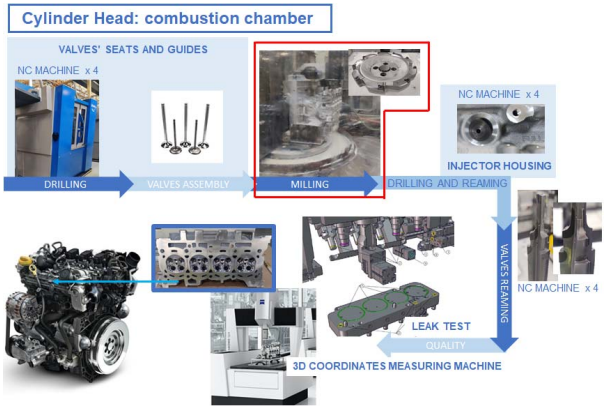

Figure 7: Manufacturing steps of combustion chambers.

The machine's lift velocity was 19m/min while the coolant was on, and the slot angle was 15 degrees. The disc was held in the lift. The stroke started and finished at the height of Z=3024mm and Z=3024mm, respectively. The machining process began at the height of Z=841mm. We recorded data from three different sources: (i) two accelerometers with a sample rate of 7.812499825377017e-05 seconds, (ii) a data logger with a sample rate of 4.00e-03 seconds, and (iii) tool wear measured with an optical microscope.

The tool wear was measured for every five slots. Two types of data were collected: average wear, which gives information about how cutting conditions are for the tool, and maximum wear, which includes micro tool breakage and is needed to replace the tool.

6.1.2 Automotive case study. It involves the data set of milling cylinder heads' combustion chambers for car engines. A combustion chamber is manufactured with several machining operations, e.g., drilling, milling, and reaming (see Figure 7). Milling is the process of machining using rotary cutters to remove material by advancing a cutter into a workpiece. A machining operation consists of cutting conditions, i.e., feed rate, depth of cut, and cutting speed.

The machining process measurements, alarms, and cutting conditions are collected directly from the CNC of each milling machine. The machines are four-axis center from Comau Urane 25 V3 – with Siemens 840D SL CNC. All information are centralized and synchronized in the KASEM application from Predict [34].

We collected around 180 times series with a sample rate between 10Hz and 100Hz. 70 time series are *analog* time series containing measurements such as axis torque or position. The other time series are machine status data, such as "machine in automatic mode". Some of the important time series obtained are (i) linear axis X, a gantry axis with two linear motors X1 and X2 (e.g., X1 and X2 real position in millimeter and X1 and X2 temperature in Celsius degree), (ii) linear axis Y and Z with only one motor (e.g., real position in millimeter and torque in percent of motor drive nominal torque), (iii) rotational axis A (e.g., position in degree and speed in degree per minute), (iv) spindle (e.g., speed in rotation per minute and current in percent of motor drive nominal current), and (v) machine condition (e.g., the number of program currently running on the machine and the number of tool load in the spindle).

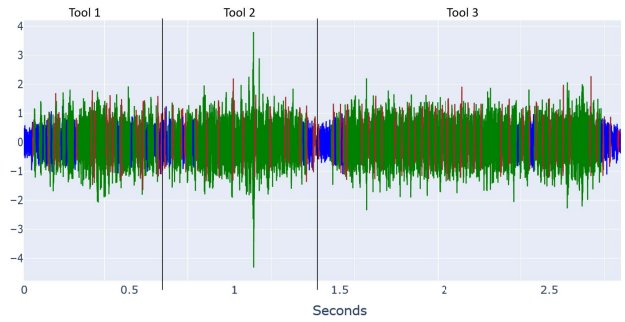

Figure 8: Aerospace case: Process behavior patterns in reference accelerometer data with tool change indications.

6.2 Results

This section discusses the results of our case studies, addressing, in turn, each of the RQs.

RQ1: Can UDAVA discover process behavior patterns in sensor data for a reference production cycle? Process behavior patterns are sections of raw sensor data in the same cluster. We obtain them from the reference data representing a successful manufacturing process.

UDAVA computed process behavior patterns in the reference accelerometer data in the aerospace case (see Figure 8). The reference data set is the vibration measurements from the milling of slot number 15 on the turbine disc (60 fir tree slots separated from each other by 4 degrees). We used this data set as a reference because (i) the broaching machine and accelerometer data is incomplete, and (ii) the data set is recorded relatively early in the broaching process when the tool has not worn out. Worn-out cutting tools can obscure the source of faults in reference data. We discovered four process behavior patterns (depicted by *four colors*) representing what the three cutting tools are doing. The red cluster is the least represented. It is attached to behavior where the tool is neither cutting nor moving. We call it the *background* cluster. We found in the visual inspection that the green and brown clusters are very similar since they are related to similar behavior and amplitudes. We call the green cluster as *cutting main* and the brown one as *cutting supplemental*. The blue cluster is related to erratic behavior where sensor data fluctuates fast and has low predictability. We have this cluster when the machine is moving but not cutting. We call it the *erratic* cluster. We found a mix of cutting and erratic clusters in the cutting behavior. These clusters are absent in non-cutting behavior. The background cluster is also present when the machine is moving. We can see it before and after the cutting process. The amplitude of both the erratic and background clusters is lower than the cutting clusters. Dimensionality reduction entails the reduction of a subsequence of raw time series data to a summary vector of a smaller size. For instance, UDAVA transforms one-hundred values over time to a 6-dimensional vector consistent with statistical properties such as mean, median, standard deviation, frequency, range. UDAVA used reduced dimensionality feature vectors to efficiently characterize behavior patterns that match a mental model of vibration states and transition between them in the broaching process.

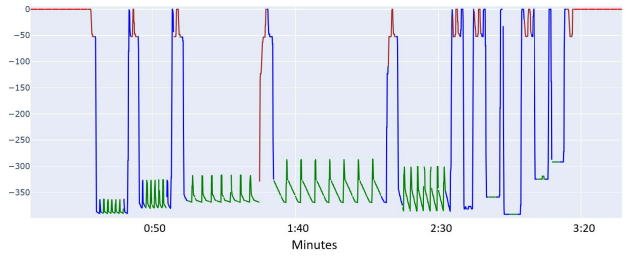

Figure 9: Automotive case: Process behavior patterns in reference spindle Z-axis position data.

UDAVA computed process behavior patterns from reference data of the machine variable Z-axis position of the spindle in the automotive case (see Figure 9). The sequence of Z-axis values is very similar and repeated in multiple batches of continuous data acquired from the milling machine. We used early batches of Z-axis values as reference data to compute clusters using UDAVA. We discovered four process behavior patterns depicted by *four colors* in the reference data. The green cluster is related to rapid fluctuation in the sensor data with low amplitude, and we call it the *oscillating* cluster. Another notable behavior pattern is the rapid vertical movement related to the blue or *vertical* clusters. Due to the quick switch between the green and blue clusters, we observed some confusion in the cluster classification where the patterns switch. We call the red one the *still* cluster since there is no movement on the Z-axis. The final cluster denoted by brown is related to the peaks, where the axis returns to baseline before instantly descending. We name it the *peak* cluster. The patterns in the time-varying Z-axis position of the spindle represent the stages of machining or CNC milling, such as power off, milling, drift, and tool change.

> ***RQ1 Conclusion.*** UDAVA assigns different behavior patterns to different clusters in a consistent fashion, allowing for unsupervised detection of various forms of behavior in the aerospace and automotive cases.

RQ2: Does comparing process behavior patterns reveal process shifts and drifts in subsequent production cycles? Comparing process behavior patterns helps visualize and quantify deviation (as discussed in Section 4.3) in multiple production cycles of a repetitive manufacturing process.

We compared behavior patterns and deviations for accelerometer data measuring vibrations during the broaching of fir tree slots around a turbine disc in the aerospace case (see Figure 10 for Slot 15, the reference slot, and Slot 45 later in the continuous broaching process). The tool wear is expected to increase with every new broached slot. The most significant difference is the deviation from the cluster centers being higher for Slot 45 (green) when compared to Slot 15 (yellow). Except for the single peak from the reference data set, all sections of the cutting process display higher averages and higher peaks. Visual inspection of the clusters in Slot 15 and Slot 45 also revealed that Slot 15 has more variance in behavior patterns when compared to Slot 45, characterized by high vibration. We present the change in behavior clusters for slots broached

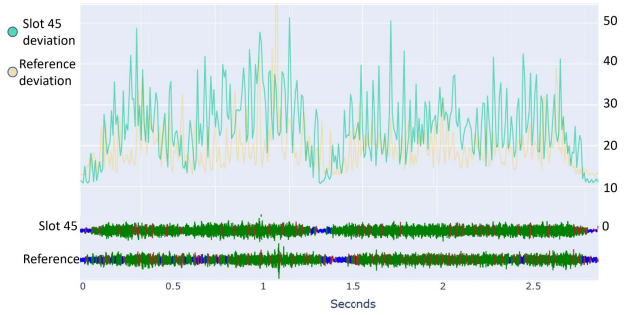

Figure 10: Aerospace case: Deviation of Slot 15 and Slot 45 from cluster centers in reference accelerometer data.

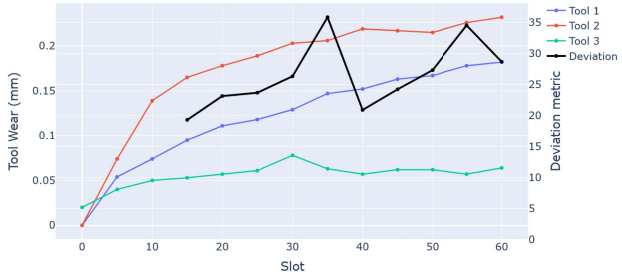

Figure 11: Aerospace case: Average tool wear per slot.

after Slot 15 (the reference) in Figure 12. A noticeable trend is a gradual decrease in cluster variation; the green *main cutting* cluster becoming more dominant indicates higher amplitudes in vibration. We compared the tool wear readings in Figure 11 with the mean deviation metric of the cutting phase from each fifth slot. We noticed a correlation between tool wear and deviation. The deviation depends on the *slot angle* (the angle at which the fir tree slot is broached on the disc). Slot angle (not in Figure 11) increases from Slot 15 to 36 and then drops. This behavior matches the drop in deviation at Slot 36 in Figure 11. We need to investigate the cause of slot angle affecting deviation. UDAVA's model as a service supports the validation of new accelerometer sensor data based on computing a simple deviation metric for each slot. Deviation in data may indicate that the sensor is not calibrated under certain conditions (e.g., slot angle) and is affected by tool wear.

We did not find a significant deviation in process behavior patterns in the spindle Z-axis position measurements in the automotive case. The spindle position data in the automotive case is sampled at a lower frequency than the accelerometer data (for vibrations) in the aerospace case. It gives a low-resolution observation of the milling process. Figure 13 compares the behavior patterns in the reference and production data. The deviation metric supports our conclusion that variation in milling was nominal with no noticeable divergence from the reference data set. We observed occasional spikes in the data set, seemingly attached to the slight difference in clustering. UDAVA's model as a service for the automotive case illustrates how the deviation metric helps validate machine sensor data, such as the Z-axis position of the spindle over many production cycles, and maintain manufacturing within expected tolerances.

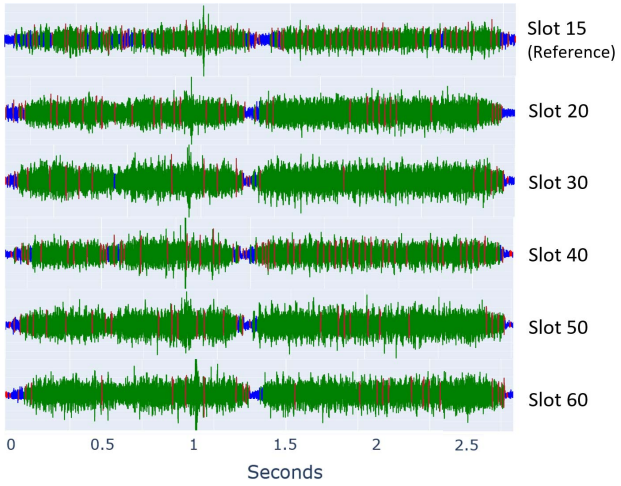

Figure 12: Aerospace case: Comparison of accelerometer data behavior patterns in many slots.

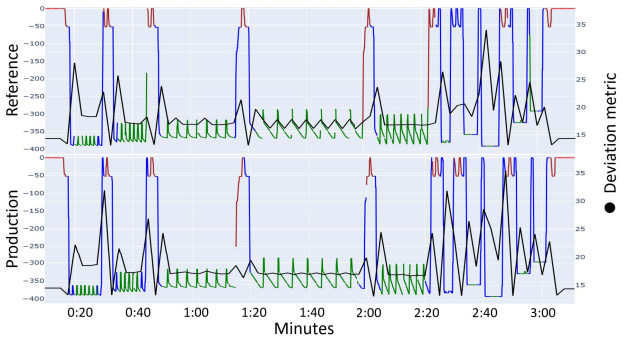

Figure 13: Automotive case: Patterns from the reference Z-axis data compared with patterns from production.

The drifts in our use cases were negligible since production cycles did not deviate much (see Figure 13). It is still possible for domain experts to set thresholds for the deviation metric in UDAVA to detect faults. One may introduce artificial faults to the manufacturing process to demonstrate significant changes in the deviation metric.

> ***RQ2 Conclusion.*** The deviation metric helps validate data over multiple production cycles and reason about the root cause of deviation by comparison with other parameters (e.g., tool wear).

RQ3: How can UDAVA be deployed, tested and maintained in industrial production environments? To address RQ3, we analyzed our experience of deploying UDAVA in two different industrial manufacturing settings. We categorized and summarized the experience based on the AI engineering dimensions in Section 2.3.

Data versioning and dependency management: Data version control (described in Section 2.1) is used in UDAVA to maintain the versions of reference data. Any change to the reference data automatically executes component dependencies to perform data preprocessing and unsupervised learning and wrap a new model as a web service (see Section 5). The reference data in the aerospace case was automatically acquired based on two data quality criteria: (a) data timeliness (earliest possible data samples with no or few signs of defects and tool wear) and (b) data completeness of internal data from the Broaching machine and external data from two high-frequency accelerometers (as depicted in Section 6.1.1. A new version of the reference data was only acquired when tools had worn out and were replaced with new tools or resharpened for reuse. The need for new reference data was manually set in a Boolean variable in the pipeline configuration. The versions of the reference data in the automotive case were obtained from an early and successful production cycle of a cylinder head of a combustion engine. The data were chosen based on a set of program numbers (referring to G-code for CNC milling) and a range of timestamps. These parameters were input into the data pipeline. The data versioning and dependency management challenges lie in the design and validation of the pipeline architecture of UDAVA. For instance, it is of interest to evaluate the bottlenecks in the performance of UDAVA given different parameters and its *greenness* through libraries such as Code Carbon [12]. Growing technical debts in data pipeline architectures [38] is also a concern for UDAVA having an increasing number of dependencies on data and code.

Deployment infrastructure: The reference data in the aerospace case was acquired from the SAVVY data systems edge device and was available as a set of CSV files for the manufacturing of each slot. Broaching a fir tree slot for a turbine disc is a repetitive and fast process. Therefore, dense high-frequency reference data over a short period from one slot was enough to validate sensor data for a new fir tree slot. In the automotive case, the pipeline obtained data through an API provided by the KASEM cloud developed by Predict to maintain a fleet of manufacturing machines. The API supports data for a specific machine, a timestamp range, and a set of variables (e.g., Z-axis position of the spindle). The deployment infrastructure presents several scientific challenges in designing new architectures, like UDAVA, on the *edge-cloud continuum* while adhering to data privacy and security constraints. As we experienced in our case studies, sensor data is the intellectual property of manufacturing companies. Therefore, many scientific challenges lie in designing federated learning architectures [19] for a fleet of machines in multi-company product lines, edge devices, and the cloud.

Quality Attributes: Data quality attributes of completeness, freshness, and accuracy are prerequisites to selecting reference data as computation of process behavior clusters, and deviation from these clusters depends on the quality of the reference data. For instance, sensor data from accelerometers measuring vibration may have missing/corrupt values due to signal processing errors such as aliasing or electrical noise from the environment recorded at the same frequency as the accelerometer signal. Data quality for reference data was validated in a different data quality service (out of scope for this article) through the specification of *expectations* using the Great Expectations tool [20]. An expectation in the aerospace case, for instance, is that 100% of the vibration severity values in the x-axis should be between 0 and 50 mm/s because a higher severity is out of range. According to an expectation in the automotive case, 95% of the spindle torque values should be within the acceptable range

of values. There were several such expectations specified for both cases. The performance of UDAVA to find meaningful clusters was evaluated based on expert knowledge of process behavior. Clustering is deemed semantically correct when UDAVA correctly clusters process behavior such as power on, normal operation, and drift. The clustering algorithm parameters (e.g., features for clustering such as frequency, mean, median, standard deviation, entropy etc.) were changed to improve performance in recognizing behaviors such as power on/off, normal operation, and sensor drift. The challenges in infusing quality attributes within a machine learning pipeline are related to streamlining user involvement in specifying expectations on data quality and evaluating clustering quality. Although UDAVA is an unsupervised learning approach, it still requires users to specify basic data quality rules and tweak clustering parameters to achieve meaningful results across multiple production cycles. Furthermore, there is a need to quantify uncertainty in clustering results due to out of distribution or uncertain data streams. Sharma et. al. [39] propose outlier-robust multi-view clustering for uncertain data to improve clustering quality in manufacturing data.

Integration of models and components: In the aerospace case, the validation of the fir tree slot was at the edge device of the SAVVY data systems. We deployed UDAVA as a simple web service in this resource-constrained edge device. The data acquired from a fleet of machines in the automotive case was transferred to the cloud having virtually unlimited resources. Here, we containerized the ML model as a service for detecting clusters and deviations in parallel on more than one CNC machine in the automotive factory. The challenges in integrating models and components are principally related to the evolution of the manufacturing process and consequently the evolution of the data and models. Although we consider the repetitive manufacturing of the same products or parts, there might be occasional, minor modifications to product specifications and process parameters (e.g., the need to ramp up production) that can render an ML model obsolete. Therefore, there is a need to investigate continual learning [30] and domain adaptation [6] in conjunction with the continuous deployment of ML models [33].

> ***RQ3 Conclusion.*** The AI engineering challenges we identified are (i) improving the greenness of pipelines, (ii) reducing pipeline debt, (iii) using federated learning in multi-machine fleets for data security, (iv) quantifying uncertainty in clustering as a quality metric, and (v) implementing continual learning, domain adaptation, and the continuous deployment of updated AI models for dynamic manufacturing environments.

6.3 Threats to Validity

Internal validity. To limit threats to internal validity, we considered the data sets obtained from assembly lines operational in factories. We included only real (physical world) data. However, limited information on changes in the production data is still a factor threatening the internal validity. The aerospace case contains tool wear information, which might explain the increase in the deviation metric, but the automotive case has no such information.

External validity. The main external threat to validity in our study concerns generalizability, which is a common issue in industrial case studies. To mitigate the threat, we considered two representative production lines manufacturing different components for different companies and generating data in different sizes and nature.

7 CONCLUSION

Manufacturing processes can generate a massive amount of sensor data at varying frequencies. It is tremendously hard for human operators to supervise such a stream of big data and detect process shifts and drifts (i.e., the signs of product defects) in the data. In this article, we presented UDAVA that discovers behavior patterns of a manufacturing process in sensor data through unsupervised learning of summary statistics. UDAVA compares the feature vectors of the production data with the cluster centers in the cluster model obtained from the reference data. It computes a deviation metric that illustrates how much the production data deviates from the original cluster centers in the reference data. The deviation metric helps to discover potential drifts in the production data, investigate the causes, and implement methods to control drifts.

We engineered UDAVA as an AI-driven system in two industrial environments: (a) broaching of turbine discs in the aerospace industry and (b) milling of cylinder heads of car engines in the automotive industry. UDAVA could identify the significant increase in the tool wear in the aerospace data set, which leads to defects in turbine discs. We also showed that it could identify process behavior patterns for normal operation and sensor drift in the automotive data set. Furthermore, we discussed how we handled some AI engineering dimensions in UDAVA for both cases.

7.1 Future work

Multi-sensor data synchronization: Multi-sensor data synchronization is mostly a manual task within the domain of prepossessing in AI. Process behavior clusters observed across multiple sensors for the same manufacturing process may have similarities in the sequence and duration of occurrence. We can use these similarities to find synchronization points between blocks of patterns across different sensors. In this respect, sensor data synchronization can be formulated as a constraint-driven optimization problem to obtain the most likely synchronization point.

AI-driven control of manufacturing processes: We plan to use the deviation metric to control the manufacturing process through tool re-sharpening, tool change, coolant use, ramp-up/ramp-down of production, and vibration compensation to reduce defects. Recommendations can be given to operators, or manufacturing process controls can be implemented automatically in real-time.

User involvement in ML-infused systems Trust in AI systems is improved when users understand the underlying model and influence it to match a mental image of the real world. Process behavior clusters detected by UDAVA do not have a label or name. Domain experts can label a few of these clusters and perform domain adaptation [50] to label all of the new sensor data automatically.

ACKNOWLEDGMENT

The work has been conducted as part of the InterQ project (958357) and the DAT4.ZERO project (958363) funded by the European Commission within the Horizon 2020 research and innovation programme.

REFERENCES

[1] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. 2015. Time-series clustering–a decade review. *Information Systems* 53 (2015), 16–38.

[2] Ali Alqahtani, Mohammed Ali, Xianghua Xie, and Mark W Jones. 2021. Deep Time-Series Clustering: A Review. *Electronics* 10, 23 (2021), 3001.

[3] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 291–300.

[4] Nagdev Amruthnath and Tarun Gupta. 2018. A research study on unsupervised machine learning algorithms for early fault detection in predictive maintenance. In *2018 5th International Conference on Industrial Engineering and Applications (ICIEA)*. IEEE, 355–361.

[5] Angelos Angelopoulos, Emmanouel T Michailidis, Nikolaos Nomikos, Panagiotis Trakadas, Antonis Hatziefremidis, Stamatis Voliotis, and Theodore Zahariadis. 2020. Tackling faults in the industry 4.0 era—a survey of machine-learning solutions and key aspects. *Sensors* 20, 1 (2020), 109.

[6] Moslem Azamfar, Xiang Li, and Jay Lee. 2020. Deep learning-based domain adaptation method for fault diagnosis in semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing* 33, 3 (2020), 445–453.

[7] Jan Bosch, Helena Holmström Olsson, and Ivica Crnkovic. 2021. Engineering AI systems: A research agenda. In *Artificial Intelligence Paradigms for Smart Cyber-Physical Systems*. IGI Global, 1–19.

[8] YUE Caixu, GAO Haining, LIU Xianli, Steven Y Liang, and WANG Lihui. 2019. A review of chatter vibration research in milling. *Chinese Journal of Aeronautics* 32, 2 (2019), 215–242.

[9] Chieh-Yu Chen, Shi-Chung Chang, and Da-Yin Liao. 2020. Equipment Anomaly Detection for Semiconductor Manufacturing by Exploiting Unsupervised Learning from Sensory Data. *Sensors* 20, 19 (2020), 5650.

[10] Yizong Cheng. 1995. Mean shift, mode seeking, and clustering. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE* (1995).

[11] Chia-Shang James Chu. 1995. Time series segmentation: A sliding window approach. *Information Sciences* 85, 1-3 (1995), 147–173.

[12] Code Carbon. [n.d.]. https://codecarbon.io/. Visited in 2022.

[13] A Del Olmo, G Martínez de Pissón, L Sastoque, A Fernández, A Calleja, and LN López De Lacalle. 2021. Merging complex information in high speed broaching operations in order to obtain a robust machining process. In *IOP Conference Series: Materials Science and Engineering*, Vol. 1193. IOP Publishing, 012079.

[14] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. 2008. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment* 1, 2 (2008), 1542–1552.

[15] Alican Dogan and Derya Birant. 2021. Machine learning and data mining in manufacturing. *Expert Systems with Applications* 166 (2021), 114060.

[16] David H Evans. 1975. Statistical Tolerancing: The State of the Art: Part III. Shifts and Drifts. *Journal of Quality Technology* 7, 2 (1975), 72–76.

[17] Tak-chung Fu. 2011. A review on time series data mining. *Engineering Applications of Artificial Intelligence* 24, 1 (2011), 164–181.

[18] K. Fukunaga and L. Hostetler. 1975. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory* 21, 1 (1975), 32–40. https://doi.org/10.1109/TIT.1975.1055330

[19] Ning Ge, Guanghao Li, Li Zhang, and Yi Liu. 2021. Failure prediction in production line based on federated learning: an empirical study. *Journal of Intelligent Manufacturing* (2021), 1–18.

[20] Great Expectations. [n.d.]. https://greatexpectations.io/. Visited in 2022.

[21] iterative.ai. [n.d.]. Open-source Version Control System for Machine Learning Projects. https://dvc.org/. Visited in 2022.

[22] ITP Aero. [n.d.]. https://www.itpaero.com/. Visited in 2022.

[23] Mustafa Kuntoğlu, Emin Salur, Munish Kumar Gupta, Murat Sarıkaya, and Danil Yu Pimenov. 2021. A state-of-the-art review on sensors and signal processing systems in mechanical machining processes. *The International Journal of Advanced Manufacturing Technology* 116, 9 (2021), 2711–2735.

[24] Ruslan Kuprieiev, Dmitry Petrov, Paweł Redzyński, Saugat Pachhai, Casper da Costa-Luis, Alexander Schepanovski, Peter Rowlands, Ivan Shcheklein, Jorge Orpinel, Fábio Santos, Aman Sharma, Zhanibek, Gao, Batuhan Taskaya, Dani Hodovic, Andrew Grigorev, Earl, Nabanita Dash, nik123, George Vyshnya, maykulkarni, Max Hora, Vera, Sanidhya Mangal, Wojciech Baranowski, Clemens Wolff, Alex Maslakov, Alex Khamutov, Kurian Benoy, and Ophir Yoktan. 2021. DVC: Data Version Control - Git for Data & Models. https://doi.org/10.5281/zenodo.4544110.

[25] Hongbin Liu, Mingzhi Huang, Iman Janghorban, Payam Ghorbannezhad, and ChangKyoo Yoo. 2011. Faulty sensor detection, identification and reconstruction of indoor air quality measurements in a subway station. In *2011 11th International Conference on Control, Automation and Systems*. IEEE, 323–328.

[26] Mika Liukkonen and Yrjö Hiltunen. 2018. Recognition of systematic spatial patterns in silicon wafers based on SOM and K-means. *IFAC-PapersOnLine* 51, 2 (2018), 439–444.

[27] Jon Loeliger and Matthew McCullough. 2012. *Version Control with Git: Powerful tools and techniques for collaborative software development.* " O'Reilly Media, Inc.".

[28] J. Macqueen. 1967. Some methods for classification and analysis of multivariate observations. In *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*. 281–297.

[29] AB Martins, JT Farinha, and AM Cardoso. 2020. Calibration and Certification of Industrial Sensors—A Global Review. *WSEAS Trans. Syst. Control* (2020), 394–416.

[30] Benjamin Maschler, Hannes Vietz, Nasser Jazdi, and Michael Weyrich. 2020. Continual learning of fault prediction for turbofan engines using deep learning with elastic weight consolidation. In *2020 25th IEEE international conference on emerging technologies and factory automation (ETFA)*, Vol. 1. IEEE, 959–966.

[31] Meinard Müller. 2007. Dynamic time warping. *Information retrieval for music and motion* (2007), 69–84.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[33] Ioannis Prapas, Behrouz Derakhshan, Alireza Rezaei Mahdiraji, and Volker Markl. 2021. Continuous Training and Deployment of Deep Learning Models. *Datenbank-Spektrum* 21, 3 (2021), 203–212.

[34] Predict. [n.d.]. https://www.predict.fr/produits-services/logiciels/. Visited in 2022.

[35] Renault Assembly Plant. [n.d.]. https://www.renaultgroup.com/en/our-company/locations/valladolid-bodywork-assembly-plant-2/. Visited in 2022.

[36] SAVVY Data Systems. [n.d.]. https://www.savvydatasystems.com/es/inicio. Visited in 2022.

[37] D. Sculley. 2010. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web - WWW '10*. ACM Press. https://doi.org/10.1145/1772690.1772862

[38] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in machine learning systems. *Advances in neural information processing systems* 28 (2015), 2503–2511.

[39] Krishna Kumar Sharma and Ayan Seal. 2021. Outlier-robust multi-view clustering for uncertain data. *Knowledge-Based Systems* 211 (2021), 106567.

[40] Carla Silva, Marvin F da Silva, Arlete Rodrigues, José Silva, Vítor Santos Costa, Alípio Jorge, and Inês Dutra. 2021. Predictive Maintenance for Sensor Enhancement in Industry 4.0. In *Asian Conference on Intelligent Information and Database Systems*. Springer, 403–415.

[41] Ashish Singhal and Dale E Seborg. 2005. Clustering multivariate time-series data. *Journal of Chemometrics: A Journal of the Chemometrics Society* 19, 8 (2005), 427–438.

[42] Dimla E Dimla Snr. 2000. Sensor signals for tool-wear monitoring in metal cutting operations—a review of methods. *International Journal of Machine Tools and Manufacture* 40, 8 (2000), 1073–1098.

[43] Hugo Steinhaus et al. 1956. Sur la division des corps matériels en parties. *Bull. Acad. Polon. Sci* 1, 804 (1956), 801.

[44] Ye Tian, Zili Wang, and Chen Lu. 2019. Self-adaptive bearing fault diagnosis based on permutation entropy and manifold-based dynamic time warping. *Mechanical Systems and Signal Processing* 114 (2019), 658–673.

[45] Laurens Van Der Maaten, Eric Postma, Jaap Van den Herik, et al. 2009. Dimensionality reduction: a comparative review. *Journal of Machine Learning Research* 10, 66-71 (2009), 13.

[46] Ethan Wescoat, Matthew Krugh, Andrew Henderson, Josh Goodnough, and Laine Mears. 2019. Vibration analysis utilizing unsupervised learning. *Procedia Manufacturing* 34 (2019), 876–884.

[47] Michael Wocker, Naomi Kimberly Betz, Christian Feuersänger, Alexander Lindworsky, and Jochen Deuse. 2020. Unsupervised learning for opportunistic maintenance optimization in flexible manufacturing systems. *Procedia CIRP* 93 (2020), 1025–1030.

[48] Zhengzheng Xing, Jian Pei, and Eamonn Keogh. 2010. A brief survey on sequence classification. *ACM Sigkdd Explorations Newsletter* 12, 1 (2010), 40–48.

[49] Hamdi Yahyaoui and Aisha Al-Mutairi. 2016. A feature-based trust sequence classification algorithm. *Information Sciences* 328 (2016), 455–484.

[50] Kaichao You, Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. 2019. Universal domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2720–2729.

[51] Jiaping Zhao and Laurent Itti. 2018. shapedtw: Shape dynamic time warping. *Pattern Recognition* 74 (2018), 171–184.