Detection of Concept Drift in Manufacturing Data with SHAP Values to Improve Error Prediction

Christian Seiffer
Business Informatics
Furtwangen University
Furtwangen, Germany
sech@hs-furtwangen.de

Holger Ziekow
Business Informatics
Furtwangen University
Furtwangen, Germany
zie@hs-furtwangen.de

Ulf Schreier
Business Informatics
Furtwangen University
Furtwangen, Germany
schu@hs-furtwangen.de

Alexander Gerling
Business Informatics
Furtwangen University
Furtwangen, Germany
gera@hs-furtwangen.de

***Abstract*—Production processes are inherently subject to dynamic change. This makes the extraction of error causes and the prediction of errors from manufacturing data by machine learning (ML) a difficult challenge, but at the same time it is the key to improve product quality and thus to economic profit. As part of the PREFERML research project (Proactive Error Avoidance in Production through Machine Learning), we present a method for detecting concept drift using clustering based on SHAP values (SHapley Additive exPlanations) and propose strategies to handle concept drift. Evaluation based on real manufacturing data shows that the cluster specific approach improves concept drift detection and can yield economic benefits.**

***Index Terms*—AI in manufacturing, error prediction, concept drift detection, clustering, SHAP values**

I. Introduction

The support provided by machine learning in the context of manufacturing processes is finding more and more application, as optimized error avoidance is a key competitive advantage. Among other things, these models can be used to predict in real time during the production process whether the examined piece of production will yield a production error at a later stage or not. A reliable prediction offers the possibility to remove individual product components from the production process that would later turn out to be defective. This leads to the avoidance of follow-up costs. However, there is a risk of incorrectly classifying error-free parts as defective, which means that future profits are not realized. A ML classification model tries to learn the underlying error-cause correlations. Yet, even if the model represents reality almost perfectly, the application can be problematic as production processes are subject to constant change. Previously learned error-cause correlations may no longer be valid in the future, which is known as concept drift. Using outdated concepts can be misleading and reduces the quality of the classification. Hence, concept drift must be recognized and the models adapted accordingly. The error-cause relationships are usually highly complex, so that there are several concepts that can be affected by concept drift to varying degrees. A blanket analysis of the entire data (without separating these concepts) does not distinguish between the individual correlations and may therefore be insufficient. SHAP values [1] allow us to subdivide the data into subsets that correspond to different concepts each. This is part of our approach by which the specific concepts can be individually examined for drift and targeted options for optimizing individual clusters can be derived. Our strategy is to monitor the quality of predictions on a cluster specific basis and to disregard predictions of errors if the expected costs exceed the expected gains.

Our work is organized as follows. Section II introduces the realities and problems of classification and concept drift in manufacturing. Section III introduces methods relevant to understanding our approach. Section IV gives an overview of the project in which this work is embedded and Section V deals with related work. Section VI describes the approach we developed. The description of the experiments in Section VII is followed by the presentation of the results in Section VIII. The paper is concluded with a brief summary and further research aspects in Section IX.

II. Domain

For a better understanding of the problem, the production environment, the application of ML as well as the problem and the handling of concept drift are explained in more detail below.

A. Manufacturing setup

A typical production setup consists of several production lines. An example of one is given in Fig. 1. It consists of a number of test stations that serve as quality gate for recent production steps. The arrangement of the test stations can be simply sequential, but also more complex. Workpieces pass through the individual test stations and are forwarded if they remain error-free. Each test station checks incoming workpieces for certain characteristics. The measurements of the individual parts are stored in the *Product Quality Management System* (PQM), that supports the monitoring of production.

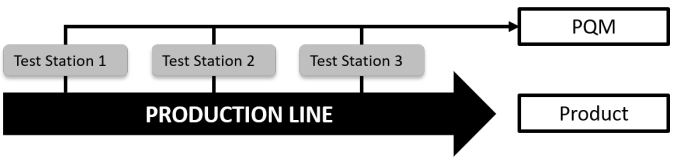

Fig. 1: Typical production setup [2]

 ISBN: 978-1-61208-891-4

If tolerance limits are exceeded during a measurement, a workpiece is labeled defective and it is removed from the production process. Products that leave the production process free of defects lead to economic profits. Early fault detection in the production process through correct error predictions therefore ensures economic savings. Defects in production can usually be traced back to causes. However, these are often complex or difficult to identify due to the large amounts of data that are generated in the PQM System.

B. Classification and prediction in manufacturing

Artificial intelligence, such as binary classification models, can be used here to work out the error-cause relationships on the one hand, and on the other hand to provide forecasts for future measurements about future susceptibility to errors. Due to the described economic implications, this has great potential. The measurement features of the individual test stations form the set of input variables for classification. The occurrence of a defective workpiece at a particular test station is stored as a binary variable, which is the target variable for classification. Its recording takes place chronologically after the measurements of input variables.

When correlations are learned through artificial intelligence, the classification models created can be used to make predictions about defects. As soon as all measured values of a certain model are acquired for a workpiece, a prediction is created. We refer to the state of a production process without support of ML as the *status quo*. Here, the workpieces pass through all production steps and are only removed from the process when an error has occurred. Alternatively, we can trust predictions of an existing ML model and remove the corresponding workpiece from the production process. Subsequent costs in production are then saved. Although the remaining production steps and test stations will not be passed, there might be possibilities to check in a separate step whether the workpiece is actually defective or not. At the same time, incorrect predictions result in missed profits when the products would actually have been error-free. It is therefore obvious to evaluate predictions in the context of manufacturing from an economic point of view. Classifications within a production line can be done in many ways. From the set of different errors that occur at a certain test station, a specific error can be selected, or a subset of all errors can be combined and treated as one fault. Any subset of the set of measurement features can be selected, provided that the measurements were made before the selected error occurred.

C. Concept drift in manufacturing

The time factor plays a major role in handling data derived from the production processes. The error-cause relationships are often not permanent, as the production environment is subject to dynamic change. For example, mechanical properties of tools used at the test stations can change and affect the tolerance limits of individual features [3]. Then learned correlations of the classification models no longer hold and there is a risk that increasingly wrong predictions are given, which is called concept drift. This effect can be detected by monitoring the quality of the positive predictions, e.g., using precision, see Section III. Each significant drift signals a change in the underlying relationships that the ML model used has not learned. If workpieces are mistakenly removed from the production process in case of a positive prediction, this yields a deterioration of the economic profit. Quality managers can provide information to determine a critical threshold at which further deterioration leads to economic losses. This state equals the *status quo* where no predictions are made. Since the quality of the predictions can only be monitored retrospectively, the decision to trust the prediction model must already be made at the previous instance. This means that a deterioration in performance can only be reacted to with a delay. Concept drift might also occur due to the increase of errors without them being correctly predicted. In this case, classification does not yield worse results than in the *status quo* and has no negative effect in comparison.

III. Fundamentals

The following sections require the knowledge of some methodological basics, which will now be presented.

A. Evaluating classification results in manufacturing

We consider the case of a binary classification model that tries to predict whether a workpiece will later be defective (class 1) or not (class 0), based on measured values in production. We evaluate results of a ML mechanism compared to the *status quo* without ML support. Not making a prediction (*status quo*) corresponds to a negative prediction of a ML mechanism. Since the positive predictions make the difference, we focus on these and neglect negative predictions.
Domingos [4] defines a cost matrix C, where $c(i, j)$ represents costs of a piece of class j that is predicted to be class i. Quality managers can assess the benefits of correct failure prediction $c(1, 1)$ and the costs of incorrect failure prediction $c(0, 1)$. They depend on the specific product, test station and error and must be determined individually. Note that $c(1, 1) > 0$, while $c(0, 1) < 0$, as there are savings when a faulty part is correctly predicted as. We define *total savings* TS as sum of savings due to the number of true positive instances TP and costs due to the number of false positive instances FP:

$$TS = TP * c(1, 1) + FP * c(0, 1) \tag{1}$$

To simplify, we set $c(0, 1) = -1$, $c(1, 1)$ can be determined as any multiple thereof:

$$TS = TP * c(1, 1) - FP \tag{2}$$

Total savings represent a metric with which the results of a classification can be evaluated from an economic point of view. As mentioned, the *status quo* does not use artificial intelligence. No errors are detected, but no workpieces are mistakenly removed from the process either, which yields $TS = 0$.

If the focus of the evaluation is on the quality of the positive predictions, then precision is a suitable metric:

$$precision = \frac{TP}{TP + FP} \quad (3)$$

In this respect, a threshold τ can be derived below which the classification leads to negative *total savings* and is therefore uneconomical.

$$TS = 0 = TP * c(1,1) - FP \rightarrow FP = TP * c(1,1) \quad (4)$$

Using (3), this gets:

$$\tau = \frac{TP}{TP + TP * c(1,1)} = \frac{1}{1 + c(1,1)} \quad (5)$$

As the precision of a single instance is a binary output, it makes sense to evaluate precision with the help of a sliding window over the last instances. In the following we call the number of instances within a sliding window n_{win}.

B. Concept drift and detecting drift in data streams

Given the quality measurements X, a ML classification model M predicts the conditional probability $P_t(Y|X)$ at time t, where Y represents a production error. We call the occurrence of the true label Y due to certain common properties of a subset of input data X *concept*. If the underlying relationships change over time, $P_t(Y|X) \neq P_{t+1}(Y|X)$ may hold after some time, which is known as concept drift. Independently of this, $P_t(X) \neq P_{t+1}(X)$ might occur as well. If there is no drift of the nature $P_t(Y|X) \neq P_{t+1}(Y|X)$, this is called virtual drift [5]. In the context of model predictions this is of smaller interest, as it does not affect the performance of classification. As the nature of the underlying classification problem is often not trivial, there might be several concepts $P_t^i(Y|X)$ that can be affected by concept drift to different degrees. This affects established performance measures of classification, such as precision or recall [5]. There are several methods for analyzing data streams for drift [6]. They have in common that they signal a deviation in data streams when it is significant, depending on various criteria. Different types of drift can be detected, such as sudden or incremental drift [5]. This complicates the quality of correct drift detection. For example, an outlier should not trigger a drift detection if there are no significant trends apart from it [6]. One of the well-known drift detectors is the Page-Hinkley test (PHT) [7].

C. State of the art drift detection in manufacturing

In order to describe the use of ML in manufacturing, we consider a point in time from which a certain amount of data is already known (Window $W1$) and from which further data can be expected in the future (Window $W2$). A typical application based on total data is shown in Fig. 2. During an initialization phase (red) a classifier model M is created based on already available data, which provides predictions for future data. This is part of the ongoing process (blue) as well as the assessment and handling of concept drift that considers all available data as a whole. All relevant information is stored within $driftLog$. Since concept drift is usually accompanied by a deterioration in prediction quality, the *status quo* with $TS = 0$ should always be considered as an alternative with regard to economic evaluation.

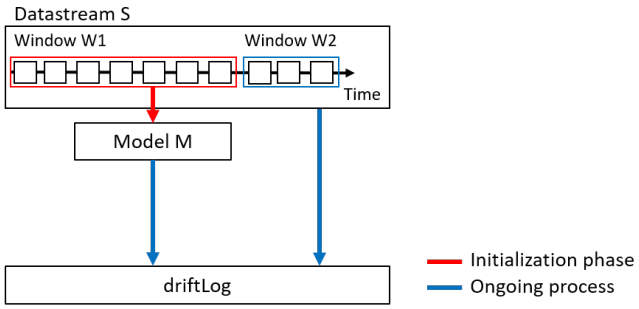

Fig. 2: Basic ML approach.

Algorithm 1 describes the procedure in detail. We define tuple t as instance of datastream S that includes the set of features X and the corresponding labels Y. Features X are derived from S by the attribute $S.X$, labels Y accordingly by $S.Y$. Model M predicts a label y for instance t. t and y are added to previous data collection $driftLog$. If y is positive, a possible concept drift needs to be investigated. The evaluation of whether a possible positive prediction of the next instance should be considered or not (boolean $ignore$) is calculated by the function $handle\_drift()$. The idea behind this is to remove workpieces when the current predictions seem reliable and not to trust them when the prediction quality is too low in order to improve the economic balance compared to the status quo. In Section VI, we propose two strategies for doing so.

D. SHAP values and their clustering

The idea of Shapley values has its origins in game theory [8]. However, the concept can be extended for the interpretation of model predictions [1]. The basis is a learned model that gives a prediction for a target variable based on a set of features for each instance. The values of each feature can influence the prediction. Depending on its value each feature contributes to the model output value of a single instance. The contribution of a feature, given that all other features remain constant, is called SHAP value. The transformation

Algorithm 1: Basic ML approach.

Input : tuple $t \in S$
classifier model M
dataframe $driftLog$
Output: dataframe $driftLog$
$y \longleftarrow M.predict(t.X)$
if y *is positive* **then**
  $ignore \longleftarrow handle\_drift(driftLog, t.X, y, args)$
$driftLog.add(t, y, ignore)$

of the original values X to the SHAP values SV will in the following be referred to as function $SHAP()$:

$$SV = SHAP(M, X) \tag{6}$$

For illustration, we trained a model with synthetic data consisting of two features A and B and 10000 instances. Feature values were generated from the standard normal distribution. 1000 randomly drawn instances were given the label *1*, which is supposed to represent an error. This adds some noise to the data. In addition, all instances with $A < -0.6$ or $B > 2.3$ received the label *1*. This created a region with significantly higher error probability. Using the model that has learned this relationship, we can create the SHAP values of an independent dataset also drawn from the standard normal distribution. Table I shows eight selected instances and the predicted error probability $\hat{y}$ calculated by the model. Values are rounded to two decimal places. The labels Y are assigned according to the learned correlations.

The prediction of the model can be understood as follows: The values of the features of Instance 1 are both outside the range with high error probability. Both features contribute to the low prediction. Instance 3 is in the critical range for A, but not for B. The model has learned that all instances in the described range are faulty. Therefore, the prediction is understandable, the value of A contributes significantly. Conversely, for Instance 5 and Instance 6, B leads to a high error probability. Both features A and B contribute to the high predictions of Instance 7 and Instance 8.

The SHAP values map this contribution of the individual features to the prediction. Table I includes the SHAP values for the named instances. Negative values signal a tendency to low error probability, positive values contribute to a high prediction. The described similarities regarding the contribution to the prediction become apparent.

Fig. 3 illustrates the SHAP values of Instance 5: The probability space $[0, 1]$ is transformed according to the logit function and forms the axis. The model output value is the prediction of Instance 5. The base value corresponds to the expected value of model output. The red arrow symbolizes feature A that contributes to an increased prediction, while the blue arrow relates to feature B that leads to a decreased prediction. The length of the arrows is the quantitative contribution and corresponds to the respective SHAP value. The value 2.61

TABLE I: EXEMPLARY INPUT DATA, CORRESPONDING SHAP VALUES, PREDICTIONS AND LABELS.

ID	A	B	$SHAP(M, A)$	$SHAP(M, B)$	$\hat{y}$	Y
1	0.39	-2.21	-1.34	-0.33	0.08	0
2	1.07	0.87	-1.28	-0.11	0.10	0
3	-0.63	-0.51	8.36	-0.02	1.00	1
4	-1.47	-0.85	8.33	-0.08	1.00	1
5	0.55	2.61	-0.74	5.56	0.98	1
6	1.16	2.34	-0.69	5.57	0.98	1
7	-0.87	2.43	6.00	2.69	1.00	1
8	-0.62	2.6	6.00	2.69	1.00	1

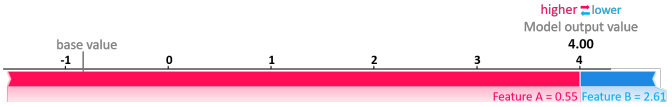

Fig. 3: SHAP values of Instance 5.

for feature B provides a significant increase of the error probability, while 0.55 for feature A speaks against an error.

If the examined instances are clustered, corresponding groups are formed which are similar with respect to their contribution to the prediction. Therefore, the assumption is that each concept $P(Y|X)$ goes along with a cluster. Since the computation of SHAP values requires a supervised learning model, the clustering of SHAP values is also called supervised clustering [9]. All SHAP values have the same unit, that of the model output. This does usually not apply to the original values of an input dataset. Thus, clustering can be done without further normalization and instances are collected whose features have a similar effect on the prediction, i.e., they are similar in terms of explanation [10]. The predictions of the instances within a cluster are always similar, but instances with similar or the same prediction can belong to different clusters. Fig. 4 shows the distribution of SHAP values from Table I. Slight jitter is added for visibility and the points are marked depending on the label of the corresponding instance. Four different clusters are clearly visible. The cluster near the origin belongs to instances where both SHAP values for A and B speak against an error. The bottom right cluster contains instances where only B tends to increase the probability of error, and the top left cluster contains instances where B is the only reason for a high probability of error. In the middle of these two clusters are instances where both A and B make an error likely.

As the number of instances increases, the main characteristics of Fig. 4 are preserved. Since the data is drawn from the standard normal distribution, clustering the input data does not provide any insights. The contributions of the features to the prediction would not be visible.

The example shows that the influence of the feature on the predictions of the individual instances is not revealed either by

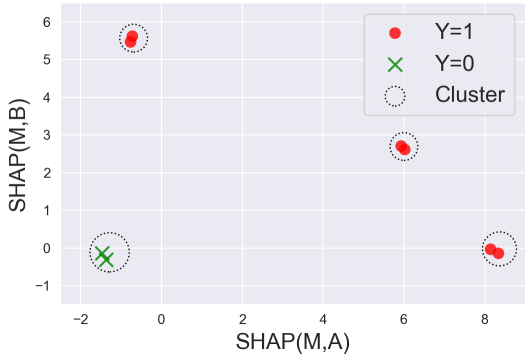

Fig. 4: Distribution of SHAP values of input data.

 ISBN: 978-1-61208-891-4

clustering the original data or by clustering instances according to their predictions.

IV. The PREFERML Concept

Avoiding errors at an early stage during production is the goal of the PREFERML project (Proactive Error Avoidance in Production through Machine Learning). The system is intended to achieve economic advantages through improved prediction quality of errors and to increase the quality of the manufactured products. To this end, the areas of machine learning processes, big data technologies and knowledge modeling are closely interlinked (see Fig. 5). The scalable application is based on the automated creation of a large number of models and their automated maintenance to reduce manual effort. The ML models generated within the framework of PREFERML serve, on the one hand, to retrospectively provide the learned error-cause correlations for quality managers. On the other hand, they provide a prediction about the defectiveness of future instances through classification. Economic benefits can only be achieved if the prediction models are of sufficient quality in the long term. To ensure this, the detection and handling of concept drift is hence a central part of PREFERML.

V. Related Work

In the following, previously published ideas and approaches related to fault detection in manufacturing and concept drift research are presented.

A. Fault detection and monitoring product quality in manufacturing

The approaches to fault diagnosis and product quality prediction include both unsupervised learning [12] and supervised learning [13]. Neural networks or decision tree-based approaches are used as classifiers in diverse applications [14], [15]. Hirsch, Reimann and Mitschang [14] compare a variety of classifiers for fault diagnosis using industrial data. They focus on the occurrence of defects in the last production step of a production line and evaluate random forests as the most suitable classifier. There are alternative ways to monitor the quality of products and workpieces during production processes. Wuest, Irgens and Thoben [16], for example, check the quality of products by monitoring the state of a product during the production process.

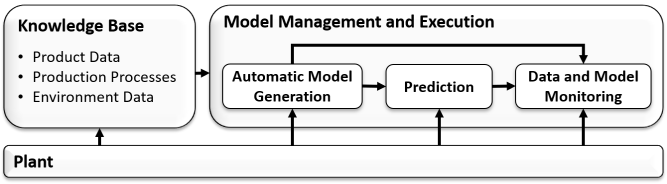

Fig. 5: Conception overview of PREFERML [11]. Arrows represent data flow between two components.

B. Concept drift research and applications

Gama, Žliobaitė, Bifet, Pechenizkiy and Bouchachia [5] shed a light on different aspects of handling concept drift and refer to state of the art methods. They discuss the various possibilities for drift detection. Besides the sequential analysis, where the Page-Hinkley test is a variant, distributions of two different time windows can be examined for statistically significant differences. With the availability of the label of incoming data, they see two alternatives: retraining or adaptation of an existing model. They define online adaptive learning as a sequence of the steps *predict*, *diagnose*, and *update*. For the evaluation of the used model the authors suggest, among others, precision and emphasize the consideration of baseline approaches. Lu et al. [17] add to this work and propose a framework for handling concept drift in the context of stream data, including *training and learning*, *prediction*, *concept drift detection*, *concept drift understanding* and *concept drift adaptation*. Accordingly, besides the time and lasting of drift occurrence, *concept drift understanding* concerns the aspects of *severity of drift* and *region of drift* (with respect to the feature space). They also see training new models and adapting existing models as consequences for dealing with drift, and propose precision as an evaluation metric, among others. They note a lack of *concept drift understanding* for existing drift detection methods, in the sense that apart from the time at which drift occurs, little information can be derived.

Adams et al. [18] try to fill this gap by expanding concept drift detection with an explainability level. Their approach introduces a cause-effect relationship to explain drifts.

Wang and Abraham [19] present an alternative approach to previously established mechanisms for detecting concept drift, which they call *Linear Four Rate (LFR)*. This method aims to identify data points that are part of the new concept to provide an updated basis for retraining. The authors show significantly better performance in terms of proven evaluation metrics compared to other methods for drift detection.

Baier, Hofmann, Kühl, Mohr and Satzger [20] present a method for handling concept drift in the context of regression problems, which they call *intersection approach*. During a period of drift predictions are derived from a simple model and a more complex model is trusted in ordinary situations.

Regression problems are also part of the work of Zenisek, Holzinger and Affenzeller [21]. They detect concept drift in industrial streaming data. However, their focus is on predictive maintenance and they look at the functionality of machines rather than a manufacturing process. They present a method in which the prediction quality is calculated within a sliding window. On this basis, a predefined threshold is used to decide whether drift is present. In a further approach, they provide a concept drift forecast on which drift detection is performed.

Just like our work, Sakamoto et al. [22] deal with concept drift detection and clustering. However, they aim to detect changes in clustering results. In our work, however, changes in classification results are the subject of investigation.

Demšar and Bosnić [23] study concept drift in streaming data using model explanation. They monitor the contributions

 ISBN: 978-1-61208-891-4

of attributes to the predictions over time. The idea is similar to our approach of using SHAP values. In contrast, we separate the individual concepts into clusters and monitor the performance of the predictions per cluster.

C. SHAP values in machine learning

Mokhtari, Higdon and Başar [24] use SHAP values in the context of financial data to obtain important features for predicting commentaries. They also show that, in their case, predictions based on SHAP values are better than predictions based on the orginal data. As part of a seismic classification task, Meng, Yang, Qian and Zhang [25] use SHAP values to determine the most important attributes. They investigate the effects of the most important attributes on the model output using SHAP plots. Lundberg, Erion and Lee [9] use an example to show the benefit of supervised clustering with SHAP values. In the context of income prediction, they identify groups with common factors relevant to income.

VI. Handling Concept Drift in Manufacturing Data

For each prediction, the question arises whether it makes economic sense to act according to it. If defects are predicted, a workpiece can be removed from the production process or it can be further processed. This decision inevitably has economic consequences: either a workpiece is correctly identified as defective and costs are saved, or the prediction is wrong and economic revenue is foregone. Our goal is to develop an approach that, based on an incoming data stream of production data, can be used to

- specifically detect concept drifts,
- efficiently derive measures and thus to
- maximize economic savings.

The decision for the upcoming workpiece has to be made on the basis of the information of all previous instances. For this purpose, we have developed a method that extends the basic use of ML in manufacturing (see Fig. 2). While the *Basic ML approach* considers all available data as a whole, our proposed method (see Fig. 6) involves dividing the same data into clusters based on their SHAP values. The initialization phase (red) remains the same, the ongoing process (blue) additionally includes the assignment of the instances to the most suitable cluster before the handling of drift.

In Subsection III-D, we showed how concepts $P(Y|X)$ can be derived on the basis of SHAP values and how the corresponding instances are distributed to clusters. We continue the example and generate 100 instances from the standard normal distribution according to those of Table I. However, we now assign the labels in such a way that a different correlation applies, namely an increased error probability if $A < -1$ (instead of $A < -0.6$ or $B > 2.3$). Adding SHAP values of these instances to Fig. 4 yields Fig. 7. Now, the actual labels are distributed differently in some clusters. The cluster near the origin with negative predictions now contains some faulty instances. For our application, these changes are less important, because they occur exactly the same with *status quo*. Of greater interest are the other clusters. Here, there are now some instances that are error-free, although faults are predicted based on certain values for A and B. In the context of manufacturing, these developments lead to consequential costs and it would be good if the error-free instances were not removed from the production process according to their prediction.

By monitoring cluster specific data we want to detect such changes of the nature $P_t(Y|X) \neq P_{t+1}(Y|X)$ and handle drift individually for each cluster. This includes a decision based on the prediction quality as to whether or not the next workpiece with a positive prediction will be removed from the production process. The detailed procedure is as follows. In a first step, all clusters are derived from the past data (here: data of $W1$), which is described in Algorithm 2. As mentioned in Subsection III-D, $SHAP(M, X)$ transforms measurements X to SHAP values according to the learned classifier M. The function $clustering()$ represents a common cluster algorithm such as k-means [26], that returns $centers$, the coordinates of the derived clusters. Each instance t of a dataframe is allocated to the closest center c of $centers$. The set $clusters$ is returned that includes all previous of these tuples. In a second step, each instance of past and future data is evaluated with regard to concept drift (see Algorithm 3). The function $handle\_drift()$ determines whether to ignore a possible positive prediction of the upcoming instance due to poor current prediction performance. We propose the following

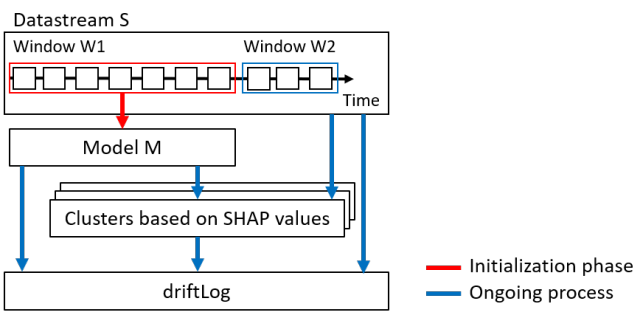

Fig. 6: Handling drift based on clustering of SHAP values.

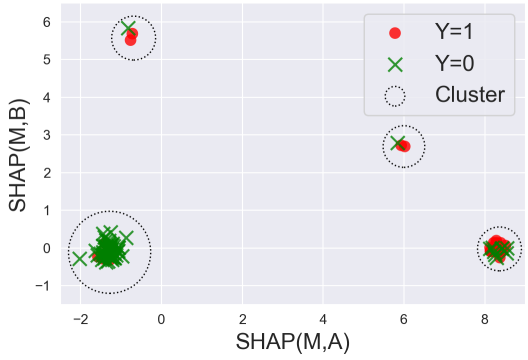

Fig. 7: Evolution of clusters based on SHAP values.

Algorithm 2: Derive clusters

Input : dataframe D
classifier model M
Output: dataframe $clusters$
$clusters \longleftarrow \emptyset$
$centers \longleftarrow clustering(SHAP(M, D.X))$
for $t \in D$ **do**
  $c \longleftarrow get\_center(SHAP(M, t.X), centers)$
  $clusters.add(t, c)$

two strategies for doing so. The variant *handle drift based on current precision only* makes this decision based on the current precision. In this case, $args$ includes the precision threshold τ and the size n_{win} of the sliding window for precision. With the actual labels Y of the input data $driftLog$ the precision of the last n_{win} positive predictions is calculated. If this is below the precision threshold τ, $true$ is returned, otherwise $false$. Due to little available data, n_{win} may exceed the number of instances so far. In this case, precision cannot be calculated and $false$ is output. This strategy is inspired by the procedure of a drift detection mechanism described by Zenisek, Holzinger and Affenzeller [21].

The alternative approach *handle drift based on current precision and established drift detection mechanisms like Page-Hinkley* extends the condition for output $true$. It requires the stream of precision values of all previous sliding windows. If the current precision is below the precision threshold τ and additionally a significant negative drift is detected for the precision stream at the current instance, $true$ is returned. This is done by established drift detectors with the help of a sensitivity parameter $sens$. $False$ is returned again, when precision is above the precision threshold τ. The second strategy is more reluctant compared to the first one, but continues to trust the prediction (possibly rightly) if the precision only briefly falls below the precision threshold τ. For this purpose, first the SHAP values of the measured values of the respective instance t are calculated and assigned to the closest cluster c via the existing cluster centers $centers$. At the same time, a binary prediction y is created on the basis of the measured values. If this is positive, all instances of the relevant cluster c are assigned to the dataframe $clusterLog$. The function $handle\_drift()$ evaluates this data with respect to concept drift and returns a boolean value $ignore$ that decides whether or not to ignore the next positive prediction. $args$ includes parameters for drift detection to be set in advance, which is described below. $driftLog$ stores the decision $ignore$ and the predicted label y of instance t.

Algorithm 3: Evaluate incoming data

Input : tuple $t \in D$
dataframe $clusters$
set of cluster centers $centers$
classifier model M
set of parameters $args$
dataframe $driftLog$
Output: boolean $ignore$
dataframe $clusters$
dataframe $driftLog$
$y \longleftarrow M.predict(t.X)$
$c \longleftarrow get\_center(SHAP(M, t.X), centers)$
$clusters.add(t, c)$
if y *is positive* **then**
  $clusterLog \longleftarrow$
   $get\_log\_of\_cluster(clusters, c, log)$
  $ignore \longleftarrow$
   $handle\_drift(clusterLog, t.X, y, args)$
$driftLog.add(t, y, ignore)$

VII. Experiments

We carried out the approach described in Fig. 6 based on real manufacturing data of our industry partner SICK AG and compared the two strategies of the function $handle\_drift$ designed in Section VI. The data used come from a total of six production lines and cover similar periods of about one year. For each experiment, a test station of a specific production line was selected. The different types of errors occurring there were combined, so that for classification an error Y always represents any type of error. All measurements recorded at previous test stations served as feature set X. This resulted in 30 experiments. These were conducted retrospectively and simulated the described methods under real-time conditions. Metadata on the data used is given in Table II.

The ML model shown in Fig. 6 resulted from the first part of the chronologically sorted data which contains 67% of the total number of errors. The remaining instances were part of window $W2$. XGBoost [27] with tree booster was used to create classification models, Page-Hinkley test served as the drift detector. Clustering was done using k-means algorithm, with the number of clusters determined by the elbow heuristic. After consulting with quality engineers, we set $c(1, 1) = 10$ and got $\tau = \frac{1}{11}$. In a pretest, a sensitivity analysis was performed on the parameters n_{win} and $sens$ (in the case of Page-Hinkley test it is called λ). A parameter setting of $n_{win} = 100$ and $\lambda = 0.1$ was considered suitable, which was chosen to conduct the experiments. For comparison, the procedure based on clustering was also performed on total data (see Subsection II-C). Note, that this resembles the state of the art of applying concept drift detection without our clustering based approach. The evaluation of an experiment is based on *total savings* TS. In the case of several clusters, these

TABLE II: METADATA ON EXPERIMENTAL DATA.

Characteristic	mean	min	max
Instances	~ 76746	10721	194932
Errors	~ 1531	139	4284
Features	~ 93	17	1105

 ISBN: 978-1-61208-891-4

are calculated by the sum of *total savings* over all clusters. Baseline for all experiments is $TS = 0$, which corresponds to the result of the *status quo*.

VIII. Results

In the following, an evaluation on the basis of a selected experiment is shown first. Afterwards, the approaches developed are evaluated on the basis of all experiments carried out.

A. Use case

The clustering in the selected case resulted in four clusters. Fig. 8 shows the occurrence of the corresponding instances over time. It includes jitter for visibility of data density. The colored area marks the corresponding time period. A vertical black line indicates the end of Window $W1$. The temporal distributions of the data of Clusters 1 and 4 are similar, as are those of Clusters 2 and 3. Noticeable are periods in which the density of the instances of Clusters 2 and 3 decreases and at the same time those in Clusters 1 and 4 increase, so that for Clusters 2 and 3 there is a longer period in which no data are assigned to them. Data of $W2$ are distributed mainly in Cluster 2. Especially in Cluster 1 it is noticeable that the density of data in $W2$ is significantly lower than in $W1$.

Since model M was created with data of $W1$ the evaluation of classification results focuses on data of $W2$. Table III summarizes the results of the classification for the clusters and total data. Especially for Cluster 2 and Cluster 3 there are negative *total savings*, which indicates concept drift.
Applying *handle drift based on current precision only* yields the results shown in Table IV. Due to the small number of positive predictions Cluster 1 and Cluster 4 are not affected. However, *total savings* for Cluster 2 and Cluster 3 can be increased clearly. Improvements are similar for total data, but slightly smaller.

This is confirmed by Fig. 9, which shows precision over time for the different clusters and total data. It illustrates precision threshold τ as black dotted horizontal line and drift detected by Page-Hinkley (λ=0.1) test as vertical red line. A vertical black line indicates the end of $W1$, y-range goes from 0 to 1 in each case. As there are too few predictions in the sliding window, no data points are available for cluster

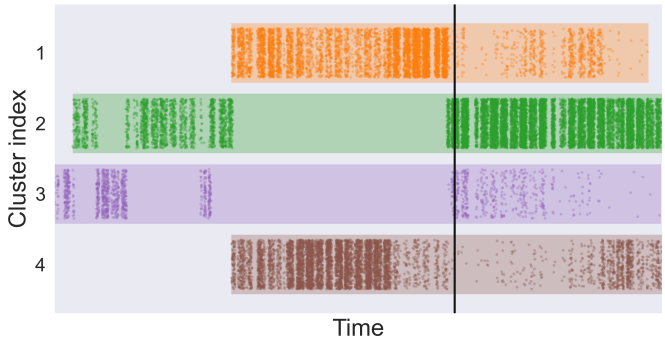

Fig. 8: Use case - time distribution of cluster instances.

TABLE III: USE CASE: CLASSIFICATION RESULTS FOR $W2$ - BASIC ML WITHOUT HANDLING DRIFT.

Data	Instances	Errors	TP	FP	TS
Cluster 1	778	16	1	0	10
Cluster 2	11094	301	139	3994	-2604
Cluster 3	479	15	15	463	-313
Cluster 4	993	34	6	0	60
$\sum_{clusters}$	13344	366	161	4457	-2847
Total data	13344	366	161	4457	-2847

1. While the precision of Cluster 2 is permanently low in $W2$, a strong deterioration of the prediction can be seen for Cluster 3 at the beginning of $W2$. In both cases, there is no lasting improvement. Cluster 2 shows the potential of a mechanism that ignores the positive predictions in case of poor performance: Once the precision is below τ, it stays there.

The strategy *handle drift based on current precision and established drift detection mechanisms like Page-Hinkley* leads to similar results for the clusters (see Table V). However, *total savings* for total data are considerably lower. With regard to the *status quo* with *total savings* $TS = 0$, it can be seen that both approaches perform better for the clusters, but the latter shows no improvement in the case of total data.

B. General view

Looking at the nature of clusters over all experiments, it turns out that some clusters have unique characteristics. For example, there are clusters whose instances are all free of errors or at least whose predicted labels are negative. Some clusters contain only instances of $W1$, so these concepts were unnecessarily learned by the model and are not used later. Yet, it may happen that only very few instances of a cluster belong to $W1$. Then only little data was available for the model to learn the corresponding concept. Accordingly, the prediction quality deteriorates in the later course. In the experiments conducted, drift occurs mostly in the form of a sudden drift, a reoccurring drift occurs only in a few cases.
The performance of the different strategies for handling drift can be determined for the individual experiments based on *total savings* of the examined case. The cluster-specific strategy *handle drift based on current precision only* ($TS = 94$) proved to be the best in the use case examined in Section VIII-A. We

TABLE IV: USE CASE: CLASSIFICATION RESULTS FOR $W2$ - HANDLE DRIFT BASED ON CURRENT PRECISION ONLY.

Data	Instances	Errors	TP	FP	TS
Cluster 1	778	16	1	0	10
Cluster 2	11094	301	22	195	25
Cluster 3	479	15	9	91	-1
Cluster 4	993	34	6	0	60
$\sum_{clusters}$	13344	366	38	286	94
Total Data	13344	366	31	307	3

 ISBN: 978-1-61208-891-4

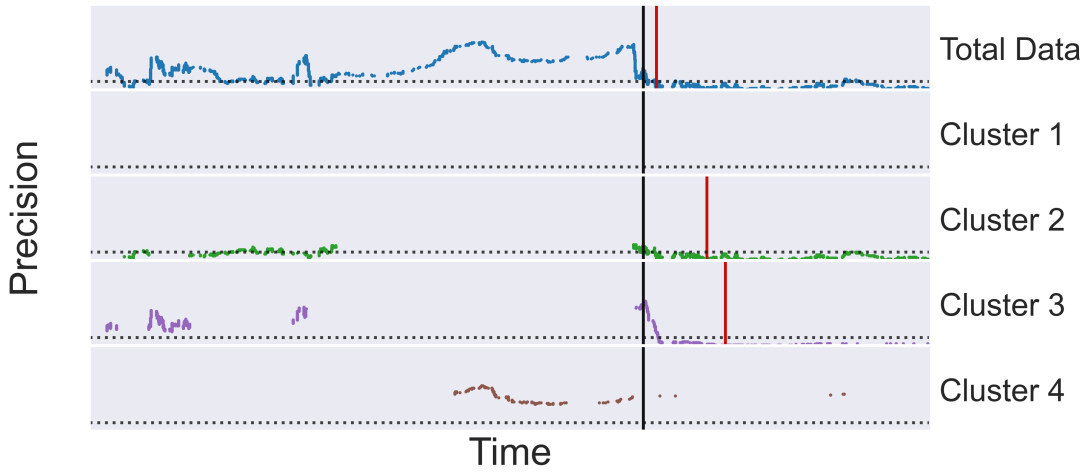

Fig. 9: Use case - cluster specific precision within a sliding window (n=100) over positive predictions.

did this evaluation for each of the 30 experiments. Summing up the *total savings* of the respective approaches across all experiments, we obtain the results shown in Fig 10.

A similar picture emerges as in the use case described: it makes more economic sense to handle drift on the basis of individual clusters. As a criterion for deciding whether a workpiece should be taken out of the production process according to a positive prediction, the current precision seems to be the most suitable. Since the *status quo* goes along with *total savings* $TS = 0$, this also applies to the totality of experiments. For a deeper analysis, the individual strategies are compared in pairs using all experiments. For each approach, we counted how often it performs best in each pairwise comparisons with all other approaches. Fig. 11 illustrates the most important pairwise comparisons.

Again, it becomes apparent that the best results are achieved with the cluster specific approaches. In a direct comparison of clustering based approaches, the strategy that handles drift on the basis of current precision only performs best. It can be concluded that precision in this application is a good metric to monitor the quality of model predictions.

IX. Conclusion

The detection of concept drift in complex relations is often difficult or does not allow detailed conclusions to be drawn. Yet, detecting concept drift is important to ensure benefits of applying a model. We have examined this challenge in the context of a manufacturing use case, where model performance impacts economic savings. We have developed and tested a method that uses SHAP values to assign the learned concepts to clusters so that they can be examined individually. Our evaluation demonstrates the benefits of the proposed clustering based approach with real manufacturing data. Here, we tested our approach for cluster specific assessment in combination with two strategies for handling drift. Our tests show better performance of drift detection using only precision values than for using the Page-Hinkley test additionally. However, in both cases our approach of clustering based assessment outperformed approaches without clustering. Note, that the

TABLE V: USE CASE: CLASSIFICATION RESULTS FOR $W2$ - HANDLE DRIFT BASED ON CURRENT PRECISION AND DRIFT DETECTION MECHANISMS.

Data	Instances	Errors	TP	FP	TS
Cluster 1	778	16	1	0	10
Cluster 2	11094	301	29	272	18
Cluster 3	479	15	10	99	1
Cluster 4	993	34	6	0	60
$\sum_{clusters}$	13344	366	46	372	89
Total Data	13344	366	47	560	-113

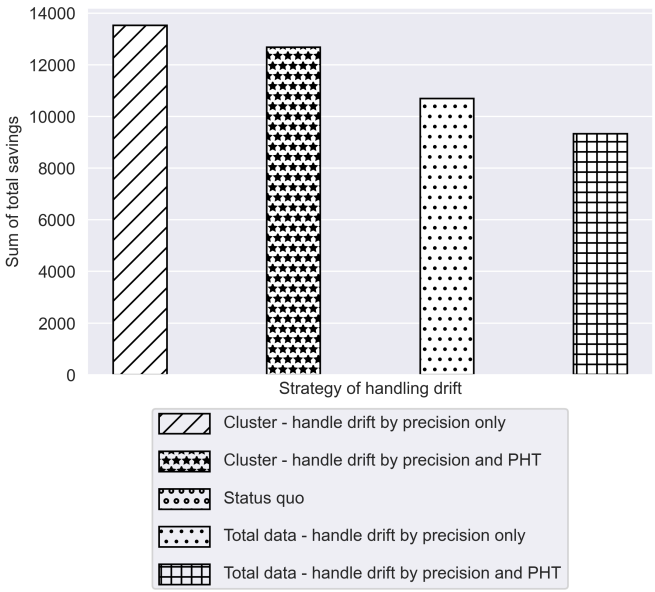

Fig. 10: Sum of *total savings* over all experiments.

 ISBN: 978-1-61208-891-4

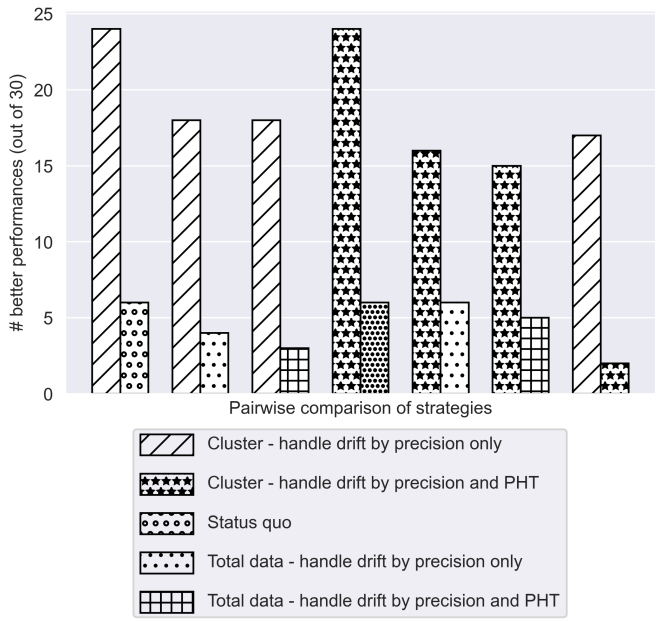

Fig. 11: Pairwise comparison of strategies to handle drift.

focus of our work is on the drift detection mechanism using clusters. Hence, we used a simple strategy for drift handling (i.e., ignoring predictions for a given cluster or total data). However, future work will address different measures such as retraining the models. The same applies to the number of the derived clusters. According to our approach, the clusters are initially inferred. It is possible that new concepts will emerge over time that actually require their own cluster. The quality of the clustering could be monitored and adjusted if necessary.

ACKNOWLEDGMENT

This project was funded by the German Federal Ministry of Education and Research, funding line "Forschung an Fachhochschulen mit Unternehmen (FHProfUnt)", contract number 13FH249PX6. The responsibility for the content of this publication lies with the authors. Also, we want to thank the company SICK AG for the cooperation and partial funding.

REFERENCES

[1] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30. Curran Associates, Inc., 2017.

[2] A. Gerling *et al.*, "A reference process model for machine learning aided production quality management," ser. Proceedings of the 22nd International Conference on Enterprise Information Systems, May 5-7,2020 : Volume 1, 2020, pp. 515 – 523.

[3] Y. Wilhelm, U. Schreier, P. Reimann, B. Mitschang, and H. Ziekow, "Data science approaches to quality control in manufacturing: A review of problems, challenges and architecture," ser. Service-Oriented Computing : 14th Symposium and Summer School on Service-Oriented Computing, SummerSOC 2020, Crete, Greece, September 13-19, 2020. Cham: Springer, 2020, pp. 45 – 65.

[4] P. Domingos, "Metacost: A general method for making classifiers cost-sensitive," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 1999, pp. 155–164.

[5] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[6] Y. Kadwe and V. Suryawanshi, "A review on concept drift," *IOSR Journal of Computer Engineering*, vol. 17, pp. 20–26, 01 2015.

[7] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[8] L. S. Shapley, *17. A value for n-person games*. Princeton University Press, 2016.

[9] S. M. Lundberg, G. Erion, and S.-I. Lee, "Consistent individualized feature attribution for tree ensembles," *ArXiv*, vol. abs/1802.03888, 2018.

[10] C. Molnar, *Interpretable Machine Learning*, 2019, https://christophm.github.io/interpretable-ml-book/, retrieved on 08/26/2021.

[11] H. Ziekow *et al.*, "Proactive error prevention in manufacturing based on an adaptable machine learning environment," ser. Artificial Intelligence: From Research to Application: The UR-AI Symposium 2019, March 13th, 2019, Offenburg, Germany. Karlsruhe: Hochschule Karlsruhe - Technik und Wirtschaft, 2019, pp. 113 – 117.

[12] N. Kolokas, T. Vafeiadis, D. Ioannidis, and D. Tzovaras, "A generic fault prognostics algorithm for manufacturing industries using unsupervised machine learning classifiers," *Simulation Modelling Practice and Theory*, vol. 103, p. 102109, 2020.

[13] T. Zhou, L. He, J. Wu, F. Du, and Z. Zou, "Prediction of surface roughness of 304 stainless steel and multi-objective optimization of cutting parameters based on ga-gbrt," *Applied Sciences*, vol. 9, p. 3684, 09 2019.

[14] V. Hirsch, P. Reimann, and B. Mitschang, "Data-driven fault diagnosis in end-of-line testing of complex products," in *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2019, pp. 492–503.

[15] K. B. Lee, S. Cheon, and C. O. Kim, "A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes," *IEEE Transactions on Semiconductor Manufacturing*, vol. 30, no. 2, pp. 135–142, 2017.

[16] W. Thorsten, D. Weimer, C. Irgens, and K.-D. Thoben, "Machine learning in manufacturing: advantages, challenges, and applications," *Production & Manufacturing Research*, vol. 4, no. 1, pp. 23–45, 2016.

[17] J. Lu *et al.*, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2019.

[18] J. N. Adams *et al.*, "A framework for explainable concept drift detection in process mining," *ArXiv*, vol. abs/2105.13155, 2021.

[19] H. Wang and Z. Abraham, "Concept drift detection for streaming data," in *2015 international joint conference on neural networks (IJCNN)*. IEEE, 2015, pp. 1–9.

[20] L. Baier, M. Hofmann, N. Kühl, M. Mohr, and G. Satzger, "Handling concept drifts in regression problems–the error intersection approach," *arXiv preprint arXiv:2004.00438*, 2020.

[21] J. Zenisek, F. Holzinger, and M. Affenzeller, "Machine learning based concept drift detection for predictive maintenance," *Computers & Industrial Engineering*, vol. 137, p. 106031, 2019.

[22] Y. Sakamoto *et al.*, "Concept drift detection with clustering via statistical change detection methods," in *2015 Seventh International Conference on Knowledge and Systems Engineering (KSE)*. IEEE, 2015, pp. 37–42.

[23] J. Demšar and Z. Bosnić, "Detecting concept drift in data streams using model explanation," *Expert Systems with Applications*, vol. 92, pp. 546–559, 2018.

[24] K. E. Mokhtari, B. P. Higdon, and A. Başar, "Interpreting financial time series with shap values," in *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering*, ser. CASCON '19. USA: IBM Corp., 2019, p. 166–172.

[25] Y. Meng, N. Yang, Z. Qian, and G. Zhang, "What makes an online review more helpful: an interpretation framework using xgboost and shap values," *Journal of Theoretical and Applied Electronic Commerce Research*, vol. 16, no. 3, pp. 466–490, 2021.

[26] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[27] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 785–794.