System Design for a Data-Driven and Explainable Customer Sentiment Monitor Using IoT and Enterprise Data

AN NGUYEN1,2, STEFAN FOERSTEL3, THOMAS KITTLER4, ANDREY KURZYUKOV2, LEO SCHWINN1, DARIO ZANCA1, TOBIAS HIPP2, SUN DA JUN5, MICHAEL SCHRAPP2, EVA ROTHGANG3, AND BJÖRN ESKOFIER1, (Senior Member, IEEE)

1Machine Learning and Data Analytics Laboratory, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 91052 Erlangen, Germany
2Siemens Healthcare GmbH, 91052 Erlangen, Germany
3Department of Industrial Engineering and Health, University of Applied Sciences Amberg-Weiden, 92224 Amberg, Germany
4infoteam Software AG, 91088 Bubenreuth, Germany
5Siemens Shanghai Medical Equipment Ltd., Shanghai 201318, China

Corresponding author: An Nguyen (an.nguyen@fau.de)

This work was supported in part by the German Research Foundation (DFG) within the framework of the Heisenberg Professorship Program under Grant ES434/8-1, and in part by Siemens Healthineers AG.

ABSTRACT The most important goal of customer service is to keep the customer satisfied. However, service resources are always limited and must prioritize specific customers. Therefore, it is essential to identify customers who potentially become unsatisfied and might lead to escalations. Data science on IoT data (especially log data) for machine health monitoring and analytics on enterprise data for customer relationship management (CRM) have mainly been researched and applied independently. This paper presents a data-driven decision support system framework that combines IoT and enterprise data to model customer sentiment and predicts escalations. The proposed framework includes a fully automated and interpretable machine learning pipeline using state-of-the-art methods. The framework is applied in a real-world case study with a major medical device manufacturer providing data from a fleet of thousands of high-end medical devices. An anonymized version of this industrial benchmark is released for the research community based on the presented case study, which has interesting and challenging properties. In our extensive experiments, we achieve a Recall@50 of 50.0% for the task of predicting customer escalations. In addition, we show that combining IoT and enterprise data can improve prediction results and ease troubleshooting. Additionally, we propose a practical workflow for end-users when applying the proposed framework.

INDEX TERMS Customer service, decision support system, IoT data, explainable AI, machine learning, big data, industrial AI.

I. INTRODUCTION

Manufacturing companies are interested in monitoring and improving their installed systems’ performance to keep their customers satisfied. The performance of a system depends on the health status of a machine and customer perception [1]. To this end, they can rely on a wealth of data collected on an ongoing basis to monitor their installed systems effectively.

Available data typically falls into two distinct groups. The first group comprises IoT data (i.e., machine logs) generated on the system, allowing one to study the problem from a machine health perspective. This is often considered in disciplines like predictive maintenance. More generally, prognostic health management [2], [3]. Sipos et al. [4] used a data-driven approach based on multiple-instance learning from event log data for predictive maintenance for high-end medical devices. For example, Calabrese et al. [5] predicted failures from event data from ball-bearing components of woodworker rover machines using tree-based classification models. Additionally, event log data are analyzed for intrusion detection [6]–[8] or failure detection in data and computing centers [9]–[11]. The second group comprises data derived from complementary enterprise systems, which enables studies from a customer perspective.
This paper presents a fully automated end-to-end machine learning framework that combines IoT (i.e., machine log) and enterprise data to model customer sentiment intending to predict escalations. We demonstrate that customer sentiment can be better estimated when looking at the system performance based on machine log data (e.g., to detect system malfunction affecting the customer) and enterprise data (e.g., ticket data from customer interactions). We use historical data of escalations as labels for our predictive models to continuously learn a probability for an escalation as an estimate for the customer sentiment. This resulting decision support system helps to prioritize customers better and troubleshoot problems. A high-level schema of the proposed framework is illustrated in Fig. 1. Furthermore, we suggest an easy and practical workflow for end-users based on our proposed decision support system framework. Additionally, we provide an anonymized industrial benchmark dataset for the research community based on the presented case study with interesting and challenging properties (i.e., multimodal, sparse and noisy labels, and concept drift) [16]. Our experimental results based on state-of-the-art machine learning methods demonstrate the effectiveness of the proposed approach. With the provided industrial benchmark dataset and source code for our experiments, we aim to encourage the research community to contribute new solutions.

The remainder of the paper is structured as follows: Section II describes the problem to be solved and the data sources. Section III describes the overall methodology. Section IV presents the experimental results. Then, section V discusses the experimental results. Following, section VI presents our proposed practical workflow. Section VII concludes our paper and discusses future research directions.

II. PROBLEM DESCRIPTION

A. BUSINESS PROBLEM

Customer satisfaction and hence service resource prioritization is a key priority in many organizations. Here, we analyze data from a large and worldwide installed fleet of high-end medical devices. Therefore, customers and local service entities can naturally differ in communicating and documenting problems due to individual expectations of system performance. This inevitably could lead to situations where customers facing similar problems address the service provider in vastly different ways. Hence, objectively prioritizing customers and service resources is a hard problem. Combining relevant information from machine logs and enterprise data could help better understand problems in the field and how they affect customer sentiment. Therefore, we design a framework for a data-driven decision support system to help to prioritize customers based on an estimated sentiment. This can help to minimize unexpected escalations as a product of more proactive customer support. Major challenges in this case study are the amount, heterogeneity, and complexity of the different data sources.

B. DATA DESCRIPTION

We use two major data sources to solve the business problem at hand, which we describe here in more detail.

1) LOG DATA

Log data is a time-based protocol of events recorded by different components of a medical system. An event consists of a timestamp (indicating when the event occurred), an event source (specifying which system component generated the event), an event id (representing a category of similar event types by the given event source), an event severity (typically: information, warning, and error), and a message text (describing the event and giving more details like sensor values). Logged events are defined and implemented by the developers of each particular system component. As a result, each developer decides the severity and amount of sensor data logged.

Depending on which combination of event source, event id, and message-text we define as unique, we get approximately $10^5$ different events. There can theoretically be an unlimited number of distinct message texts depending on the usage. One system typically generates from $10^4$ to $10^6$ of these events per day. A typical system family with several thousand installed systems worldwide can generate up to 100 GB of log data per day.

Customer support centers typically use these log files to diagnose problems and the original system developers to track whether their developed systems work as intended. Major challenges for analyzing log data are volume and complexity.
2) ENTERPRISE DATA SOURCES

Enterprise data sources are mostly collected by and stored in enterprise resource planning (ERP) systems. Types of enterprise data are:

- service activity data/ticket data - documenting all customer interactions and problems which occur
- spare part data - typically related to service activities, include which spare parts have been used for maintenance/repair of a system
- customer base/contract data - listing all customers and the corresponding relationships, especially what kind of service level has been signed

Major challenges regarding the enterprise data are:

- getting a consistent picture for all customers and service activities worldwide, which is made more difficult by different local ERP systems
- manual data inputs contain errors due to typos and incorrect usage
- worldwide standards differ a lot, especially since there is no precise definition of what a “well running medical system” is and, therefore, interpretation of service data can differ from country to country
- regional ticket data is often written in the local language

Globally operating companies can have several customer service centers ranging from regional to global, and all of them are generating ticket data. In our case, we consider three different ticket levels, from regional to global. Furthermore, we analyze tickets generated by an information system tracking escalations from customer service to the R&D department.

C. REQUIREMENTS

Special requirements have to be fulfilled for a successful deployment of a decision support system. We describe these requirements in this section and adapt all design decisions accordingly. During the whole development life cycle from proof-of-concept to deployment, we worked closely together with domain experts and stakeholders from all relevant departments, including potential end-users. Thus, we could ensure that we fulfilled all requirements and built a framework that has a real impact on the end-users.

- **Dynamics and Efficiency**: Currently, decisions about escalations are made weekly. Hence, the overall framework should efficiently load new data, extract features, train a model, and perform predictions weekly. The framework should be able to do this within a few hours, e.g., on Monday mornings.
- **Model Performance and Output**: The escalation flags (highest escalation level) used as labels in this case study are very sparse and noisy. This causes special challenges for the prediction task. A single binary output is not desired, but rather the probability of escalation, which models the customer sentiment. Customers with the largest probabilities will then be analyzed in more detail by the end-users. It is not the main goal to design a machine learning model which perfectly predicts escalations. Rather, the decision support system should help identify, based on the designed features, which customers might need special attention.
- **Interpretability**: This is especially important for real-world applications as the present case study. The end-user would like to understand the reasoning behind the prediction model’s output to take the appropriate actions and build trust. For this reason, we extract specific features from log and enterprise data. These features were designed with domain experts and end-users to incorporate prior knowledge and interpretable features into the decision support system.
- **Usability**: We developed an interactive application based on a commercial Business Intelligence tool that is currently in use by the medical device provider. Usability also includes considerations of what data sources need to be provided based on the end-user’s request. The ability to interact with the provided decision support system enables continuous feedback for validation. Based on the explanations of the model and features, the end-user can decide if the predictions are valid and provide more and cleaner labels for future prediction cycles. We will later describe an envisioned workflow for the designed decision support system.

III. METHODOLOGY

In this section, we describe the designed framework to solve the business problem at hand. A high-level overview of the implemented framework is depicted in Fig. 1. Hundreds of gigabytes of incoming log and enterprise data worldwide are automatically analyzed via a log evaluation framework to calculate relevant features designed with domain experts. Based on that, we design an automated and interpretable machine learning pipeline to calculate a probability of escalation (customer sentiment), which models the customer sentiment. A complementary designed decision support system explains the predicted probability and historical data based on extracted features from log and enterprise data for the end-user. There are major benefits when combining both data sources from the end-user perspective. Depending on which features explain a prediction, we can identify related to R&D (log features) or customer service (enterprise features).

A. BUILD DATASET

Algorithm 1 describes how we built the dataset for the experimental setup. In the following, we will describe the process in more detail.

1) LABELING

Let I be the set of all customers. Fig. 2 depicts the labeling approach for one exemplary customer \( i \in I \) using a sliding window approach. The step size is set to 1 week since decisions about escalation are made weekly as described in subsection II-C. We set the window size to 10 steps, which domain experts proposed. Different values were also
FIGURE 1. High-level schema of the proposed framework. Systems from all over the world are sending log data. Additionally, enterprise data (sales and ticket data) are collected. Features are extracted based on domain knowledge to train a prediction model with historic escalation data as labels. The resulting data and predictions are integrated into a decision support system for the end-user. The framework is general and independent of specific data transmission (e.g., 5G or cloud) and storage (e.g., cloud or on-premise) technologies. Furthermore, the prediction model can be freely chosen (e.g., Random Forest or Neural Network). The decision support system can be an interactive graphical user interface (e.g., dashboard).

evaluated but did not yield an improvement. From this window, a feature vector \( x_{i,\text{pred}} \) is extracted, where \( t_{\text{pred}} \) is defined as the last week in a window.

Let \( T_{i,\text{esc}} \) be the set of all escalations flags for customer \( i \) and \( t_{\text{esc}} \) a specific time point of an escalation flag for customer \( i \). The predictive interval is set to 2 steps. If there was an escalation \( t_{i,\text{esc}} \in T_{i,\text{esc}} \) in the predictive interval, the label \( (y_{i,\text{pred}}) \) for this sample is set to 1 and 0 otherwise (line 9-12).

After an escalation \( t_{i,\text{esc}} \), the following 4 steps are defined as an infected interval. Domain experts chose this value since we assume that there is already a special focus on customers for which a recent escalation occurred. All samples where the sliding window contains weeks from the infected interval are excluded (line14-15).

As described in line 2-3, we repeat this procedure for all customers for a fixed time frame of 104 steps, which in our case is equivalent to 2 years. With that, we can simulate the real-world performance of our framework for one full year (one year of data is used for training). More details regarding training and validation will be explained later. The resulting dataset \( D_{\text{pred}} = (x_{\text{pred}}, y_{\text{pred}}) \) contains samples for all customers with complete data (line 4). The resulting distributions are depicted in Fig. 3. Note that the number of customers \( |y_{\text{pred}}| \) is increasing while the number of escalations \( ||y_{\text{pred}}||_1 \) remains almost constant over time. This is due to limited service resources, limiting the number of customers put into focus each week.

Finally, we provide an anonymized version of \( D \) for the research community as an industrial benchmark dataset [16].

2) FEATURE EXTRACTION
Feature extraction was required due to the high volume and complexity of existing data sources. Weekends and public holidays can introduce noise into calculated features. Therefore, we decided together with domain experts to aggregate features weekly. For all customers \( i \in I \) and prediction weeks \( t_{\text{pred}} \), we calculate features for a window of 10 weeks (line 2-8).

Log Data: As described subsection II-B1, the machine log data is not feasible to analyze in their raw format. Instead, we use a log evaluation framework to detect specific sequences of events determined by domain experts. The extracted features have clear meanings related to specific system malfunctions affecting customers’ daily work routines. Such features include, for example, abort of operation, system delay, user interface (UI) freeze, and UI pop-ups. We also extract whether there was a software (SW) update performed for a system.

Enterprise Data: As described in subsection II-B2, the enterprise data available can be split into two connected groups - sales data and customer service tickets. Features for sales data are the number and total cost of replaced parts. Features for sales data are the number and total cost of replaced parts. Features derived from ticket data include the number of open tickets, the age of the oldest open ticket, the rated severity, and the number of site visits for each customer, depending on availability in the different ticketing systems. These features can be extracted on a global level.
A. Nguyen et al.: System Design for Data-Driven and Explainable Customer Sentiment Monitor

FIGURE 2. Description of the labeling approach for one example customer $i \in I$. We assume that $t_{\text{pred}}$ (last week in window) starts at some point $t_0$. (a) Sample with negative label ($y_i, t_{\text{pred}} \leftarrow 0$) since there is no escalation within predictive interval of 2 weeks. (b), (c) Samples are labeled positive ($y_i, t_{\text{pred}} \leftarrow 1$) since there is an escalation within the predictive interval. (d) We exclude infected samples from the dataset. (e) First valid sample after the infected interval.

![Figure 2](image)

FIGURE 3. Sample distribution for the whole dataset $D$. The graph depicts all samples ($|y_{t_{\text{pred}}}|$) and positive samples ($\|y_{t_{\text{pred}}}\|_1$) for $t_{\text{pred}} = t_0 + \{0, \ldots, 104\}$.

3) TIME COMPLEXITY
Algorithm 1 has worse case time complexity $O(n^3)$ since we have 3 nested loops depending on: the number of customers $|I|$, the number of time steps $|W_{\text{exp}}|$ to consider, and the number of time steps of the observation window $|W_{\text{obs}}|$. Note that the time complexity is not of concern for the case study since we only need to update new data once per week.

B. PREDICTION MODELS
To meet the requirements described in section II-C, we selected a specific approach to model the customer sentiment. Therefore, a binary classification problem was formulated for the dataset $D$. Then, we model the customer sentiment as the probability for escalation based on the output of the binary classification model. The output of an interpretable machine learning model can be helpful in several ways. First, we can prioritize customers based on the predicted customer sentiment (high probability of escalation). Additionally, we can analyze which specific problems depicted in the designed features might lead to escalation. Note that there is no comparable benchmark dataset available in the literature to the best of our knowledge. We compare different state-of-the-art machine learning methods: ensembles of decision trees and deep neural networks (DNNs) and open-source our data set to provide a novel benchmark to the research community. For both methods, we calculate post-hoc explanations for each prediction using either a tree explainer [17] or a modification of DeepLift, [18]. Both algorithms are implemented in the SHAP library [18], and we refer to the explanatory outputs as SHAP values.

1) ENSEMBLE OF DECISION TREES
Ensemble of decision tree methods have the following benefits:
- The computed feature importance [19], [20] can help end-users understand which of the designed features are “correlating” with escalations/customer sentiment.
- Ensemble methods provide a probability as a model output which can be interpreted as the customer sentiment (probability for escalation).
- Since each combination of time point (week) in a window and designed feature is modeled as a single input variable, we can provide the relevance of each input variable for all predictions to the end-user for better troubleshooting.

The decision tree ensemble methods we select are Random Forest (RF) [21] and XGBoost (XGB) [22]. Random Forest and XGBoost are ensemble learning techniques that can be used for both classification and regression. In our case, we are interested in a binary classification problem.

In general, an ensemble learning technique is a collection of weak classifiers $\{C_i\}$ where each base classifier gets the same input $x_{t_{\text{pred}}}$ and outputs the predicted class $C_i(x_{t_{\text{pred}}}) \in \{0, 1\}$. The probability output of the ensemble algorithm is defined as follows:

$$\hat{y}_{t_{\text{pred}}} = \frac{\sum_{n=1}^{N} \hat{y}_{t_{\text{pred}},n}}{N} \in \mathbb{R}[0,1]$$  

2) https://github.com/slundberg/shap
3) https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
4) https://xgboost.readthedocs.io/en/latest/python/python_api.html
The decision trees for RF are generated independently and in parallel via a bagging (bootstrap aggregation) approach [21]. The objective function to be minimized here is the Gini impurity criterion. This means that each decision tree is generated in two steps:

1) Bootstrapping: Independently sampling the input data set $D_{\text{train}} = (x_j, y_j)$ with $j \in \{1, \ldots, m\}$ for each base classifier $C_i$ on data points and features.

In other words, the data points in $D$ are sampled i.i.d. (independent and identically distributed) into a subset $D_{\text{train}} = (x_j, y_j)_{j \in J_i}$ where $J_i \subset \{1, \ldots, m\}$.

Also, the feature space is sampled i.i.d., such that if $x_j$ contains the features $F = \{f_k : k \in \{1, \ldots, n\}\}$, then $x_j$ contains the features from $F^i \subset F$.

2) Aggregating: Averaging or in our case deciding by a majority vote which class should be chosen. In our case we are interested in the probability output.

Gradient Boosting [23] also combines many weak classifiers into a strong classifier. In contrast to bagging, the decision trees are not built in parallel but sequentially, while results are combined along the way. In our case, we applied XGBoost [22] as a gradient boosting framework.

In both cases, the model outputs the predicted class (here: 0 or 1) and the probability the model assigns to each prediction. We use the probability the model assigns to class 1 as the predicted customer sentiment. $\hat{y}_{\text{pred}} \in \mathbb{R}_{[0,1]}$ for each input $x_{\text{pred}}$. Fig. 4 depicts how an ensemble of decision trees produces a prediction based on a majority vote.

![Majority vote schema for an ensemble of decision trees.](image)

We address the imbalanced class problem by applying either random oversampling of the minority class, SMOTE, [24] or random undersampling of the majority class [25] using the imblearn library [26]. We treat the sampling strategy as a hyperparameter in our model selection approach, which we will describe later. We tested 8 different model configurations, as summarized in Table 1. We applied two different data fusion approaches. For “early” fusion (M1, M2) we simply stacked enterprise ($x_{\text{ent}}$) and log ($x_{\log}$) features to train a single classifier (RF or XGB). For “late” fusion (M3, M4) we trained one ensemble classifier based on $x_{\text{ent}}$ and one based on $x_{\log}$. The output of each ensemble classifier was then fed into a subsequent logistic regression layer for the final prediction. Both ensemble classifiers are either RF or XGB. We additionally tried to train a classifier only based on $x_{\text{ent}}$ (M5, M6) or only on $x_{\log}$ (M7, M8).

2) DEEP NEURAL NETWORKS

We implemented a DNN based on long short-term memory (LSTM) neural networks (NNs) [27] in order to model $x_{\text{pred}}$ as a time series. Given a sequence of inputs $x_{\text{pred}} = (x^{(1)}, x^{(2)}, \ldots, x^{(10)})$, a LSTM computes sequences of outputs $(h^{(1)}, h^{(2)}, \ldots, h^{(10)})$ via the following recurrent equations:

$$f_g^{(t)} = \sigma(W_f[x^{(t)}, h^{(t-1)}] + b_f)$$

$$i_g^{(t)} = \sigma(W_i[x^{(t)}, h^{(t-1)}] + b_i)$$

$$c^{(t)} = \tanh(W_c[x^{(t)}, h^{(t-1)}] + b_c)$$

$$o_g^{(t)} = \sigma(W_o[x^{(t)}, h^{(t-1)}] + b_o)$$

$$h^{(t)} = o_g^{(t)} \circ \tanh(c^{(t)})$$

\[\forall t \in \{1, 2, \ldots, 10\}.\]  

\[\{W_{f,i,c,g,o}, b_{f,i,c,g,o}\}\] are trainable parameters, $\sigma$ is the sigmoid activation function, $\circ$ denotes the Hadamard product (element-wise product), $h^{(t)}$ and $c^{(t)}$ are the hidden state and cell memory of a LSTM cell. Additionally, an LSTM cell uses four gates to update its states to avoid the problem of exploding/vanishing gradients in the case of longer sequences [28]. $f_g^{(t)}$ (forget gate) determines how much of the previous memory is kept, $i_g^{(t)}$ (input gate) controls the amount of new information ($c^{(t)}$) stored into memory, and $o_g^{(t)}$ (output gate) determines how much information is read out of the memory. The hidden state $h^{(t)}$ is commonly forwarded to a successive layer. In our experiments, we set the number of LSTM layers to $\ell \in \{1, 2\}$ as a hyperparameter. Additionally, we set as a hyperparameter if the LSTM is bidirectional [29] or not. The final output vector from the last LSTM cell $h^{(10)}$ is then forwarded to a fully connected layer using dropout [30] for regularization and softmax activation for prediction. Fig. 5 depicts the implemented DNN architecture. For data preprocessing, we applied standardization of the input data. Therefore, we scaled all input features by the mean and standard deviation of the training data. As an objective function, we used the binary cross-entropy loss to train the network. To address the imbalanced class problem, we applied either random oversampling of the minority class or random undersampling of the majority class [25]. We used the PyTorch framework \(^5\) to implement our DNN architecture.

\(^5\)https://pytorch.org/
Furthermore, we define \( N \) where performance of our decision support system. Algorithm 2

Our experiments are designed to simulate the real-world

FIGURE 5. Implemented DNN architecture.

C. TRAINING AND VALIDATION

Our experiments are designed to simulate the real-world performance of our decision support system. Algorithm 2 line 1-9 and Fig. 6 depict the experimental setup.

1) EVALUATION METRICS

For a given set of estimated customer sentiment values \( \hat{y}_{t_{\text{pred}}} \) and ground truth escalation labels \( y_{t_{\text{pred}}} \), we calculate the Recall@N for a whole year \( t_{\text{pred}} = t_0 + \{53, \ldots, 104\} \) as an evaluation metric. This evaluation metric reflects the practical application of our framework. End-users will usually scan customers with the \( N \) largest estimated customer sentiment (risk of escalation) weekly. The goal is then to identify and prevent as many potential escalations as possible by doing so. \( \|y_{t_{\text{pred}}}\|_1 \) is the number of positive samples and \( \|y_{t_{\text{pred}}}^{\text{pred}}\| \) the total number of samples at \( t_{\text{pred}} \), respectively. We define

\[
\text{Recall@N} = 100 \cdot \frac{\sum_{t_{\text{pred}}} \text{True}(N(\hat{y}_{t_{\text{pred}}})))}{\sum_{t_{\text{pred}}} \|y_{t_{\text{pred}}}^{\text{pred}}\|_1},
\]

where \( N() \) denotes the \( N \) largest elements and True() denotes the number of samples which have a positive ground truth value. Furthermore, we define

\[
\text{avg(Recall@N)} = \frac{\sum_{N=1}^{100} \text{Recall@N}}{100},
\]

as the average Recall@N, in order to compare different models for a relevant range of values \( N \). For both metrics the worst and best possible values are 0 and 100 respectively.

2) MODEL SELECTION AND EVALUATION

We performed a weekly analysis for one year (\( t_{\text{pred}} = t_0 + \{53, \ldots, 104\} \)) to evaluate our approach. For each week, we used the past year as training data (\( D_{\text{train}} \leftarrow D_{\text{pred}} - 53:t_{\text{pred}} - 2 \)). The gap of 2 weeks is needed since in deployment we would have complete data until \( t_{\text{pred}} \). Therefore, we only know the label for samples until \( t_{\text{pred}} - 2 \), given our predictive interval is 2 weeks. Fig. 7 shows the resulting distributions for \( D_{\text{train}} \) and \( D_{\text{test}} \). We additionally split the training data for model selection into the first 41 (\( D_{\text{train}^*} \leftarrow D_{\text{pred}} - 53:t_{\text{pred}} - 13 \)) and last 10 (\( D_{\text{val}} \leftarrow D_{\text{pred}} - 11:t_{\text{pred}} - 2 \)) weeks.

For hyperparameter tuning, we applied the Tree-structured Parzen Estimator (TPE) [31] approach. TPE is a Bayesian optimization approach for hyperparameter tuning and can yield better results than grid search and random search [31]. We used the TPE implementation [32] library for our pipeline. We trained models with different hyperparameter combinations suggested by TPE on \( D_{\text{train}^*} \) and calculated the \( \text{avg(Recall@N)} \) for \( D_{\text{val}} \).

The LSTM model (M9) was trained for 150 epochs on \( D_{\text{train}^*} \) and stopped training if the validation loss did not decrease for 15 epochs. A model checkpoint with the minimum validation loss was chosen to calculate the \( \text{avg(Recall@N)} \) on \( D_{\text{val}} \). Finally, the LSTM model with minimum \( \text{avg(Recall@N)} \) on \( D_{\text{val}} \) over all sampled hyperparameter combinations was chosen for final evaluation on \( D_{\text{test}} \).

For RF and XGB, we used the best set of hyperparameters to train a model on the complete training data \( D_{\text{train}} \). The resulting model was used to calculate predictions on the current test data (\( D_{\text{test}} \leftarrow D_{\text{pred}} \)) in order to obtain the estimated customer sentiment \( \hat{y}_{t_{\text{pred}}} \). The hyperparameters for the different classifiers are listed in Appendix VII.

For evaluation, we calculated Recall@N based on \( \hat{y}_{t_{\text{pred}}} \) and \( y_{t_{\text{pred}}} \) over the whole year (\( t_{\text{pred}} \in \{t_0 + 53, \ldots, t_0 + 104\} \)). This measures the percentage of escalations we would have predicted in one year if we would look at the \( N \) largest estimated customer sentiments at each week.

In deployment (Algorithm 3), we provide information regarding the customer sentiment in the current week (\( t_{\text{pred}} + 1 \)) based on \( \hat{y}_{t_{\text{pred}}} \) since we only have the full data available up to and until \( t_{\text{pred}} \).

The source code for our experiments is available on GitHub.\(^6\)

3) TIME COMPLEXITY

Algorithm 2 has time complexity \( O(n^2) \cdot O_M \), where \( O_M \) is the time complexity of the model training and testing. The term \( O(n^2) \) is due to the loops over the simulated weeks and trials for the hyperparameter tuning. Note that the time complexity

\(^6\)https://github.com/annguy/customer-sentiment-monitor
is not of concern for the case study since we only need to update the models once per week.

**FIGURE 6.** Illustration of the proposed training and evaluation setup (Algorithm 2). We evaluate the decision support system on a weekly basis for one year ($t_{\text{pred}} \in \{t_0 + 53, \ldots, t_0 + 104\}$). For each week, we use the data from the previous year (52 weeks) to train a model and to get a probability output for each customer ($\hat{y}_{t_{\text{pred}}}$). Finally, we calculate Recall@N to evaluate the performance over the whole year.

**FIGURE 7.** Resulting distribution of samples in $D_{\text{train}}$ and $D_{\text{test}}$. The graphs depict all samples ($|y_{t_{\text{pred}}}|$) and positive samples ($\|y_{t_{\text{pred}}}\|_1$) for $t_{\text{pred}} = t_0 + \{53, \ldots, 104\}$.

**IV. RESULTS**

Figure 8 shows the Recall@N values over $N \in \{1, \ldots, 100\}$ for the different model configurations (Table 1). Additionally, Table 2 depicts specific Recall@N values for $N \in \{10, 20, 30, 40, 50, 100\}$ and the overall avg(Recall@N).

**Early and Late Fusion:** Comparing M1 vs. M3 and M2 vs. M4 shows that there is a slight overall benefit of late fusion compared to early fusion in terms of avg(Recall@N) (36.89 vs. 39.07 and 16.36 vs. 13.97). The only exception where XGB performs better than RF is the configuration with enterprise features only (M6 vs. M5) in terms of avg(Recall@N) (34.11 vs. 35.99).

**Deep Neural Networks:** Our LSTM model (M9) is consistently outperformed by the other models using both feature sets (M1-M4) in terms of avg(Recall@N) (34.43 vs. 36.89 – 44.13).

**TABLE 1.** Different model configurations used for the experiments.

| Model | features | classifier | fusion |
|-------|----------|------------|--------|
| M1    | $x_{\text{log}}$, $x_{\text{ent}}$ | XGB    | early  |
| M2    | $x_{\text{log}}$, $x_{\text{ent}}$ | RF     | early  |
| M3    | $x_{\text{log}}$, $x_{\text{ent}}$ | XGB    | late   |
| M4    | $x_{\text{log}}$, $x_{\text{ent}}$ | RF     | late   |
| M5    | $x_{\text{ent}}$             | XGB    | n.a.   |
| M6    | $x_{\text{ent}}$             | RF     | n.a.   |
| M7    | $x_{\text{log}}$             | XGB    | n.a.   |
| M8    | $x_{\text{log}}$             | RF     | n.a.   |
| M9    | $x_{\text{log}}$, $x_{\text{ent}}$ | LSTM   | early  |

**V. DISCUSSION**

Model configurations M2 and M4 yielded the best results (RF with early and late fusion). Thereby, late fusion (M4) performed slightly better in terms of avg(Recall@N) (42.46 vs. 44.13). In practice, it is harder to compute meaningful SHAP values for the late fusion approach since both ensemble classifiers are trained on different feature sets. Therefore, our practical recommendation is to use M2 with early fusion. RF generally performed better than XGB. The deep learning-based approach M9 achieved inferior results compared to the best-performing tree-based models. We observed during our experiments that XGB and deep learning-based models were more prone to overfit the training data than RF. Furthermore, XGB and LSTM based models are more than 10 times slower to train than RF and are harder to tune. We also noticed that the computation of SHAP values [18] for LSTM based models is significantly slower than XGB and RF. Further research is needed to improve the performance.
of these models. In conclusion, we recommend using model configuration M2.

In practice, we estimate that one customer support employee can scan around 10 customers in-depth with our framework per hour. Hence, if, for example, a team of 5 customer support employees would invest one hour each week using the proposed decision support system with model M2, they could potentially prevent around 48.61% of escalations in a year. Furthermore, the decision support system can help learn which specific problems, depicted in the designed features, lead to escalation and identify customers who have similar problems and might need special attention. In the following section VI, we explain how the designed decision support system could be used in practice.

VI. PROPOSED WORKFLOW

The following section briefly outlines how the data-driven decision support system can be integrated into a productive environment and how customer support employees could use it.
FIGURE 8. Recall@N curves for $N \in \{1, 2, \ldots, 100\}$ for all model configurations listed in Table 1.

FIGURE 9. Feature importance for all trained models for configuration M2.

The envisioned workflow, which is how this system is used, can be grouped into five steps outlined in Fig. 10.

- **Step 1: Producing new predictions** This step is fully automated, and all relevant processes are triggered at the beginning of each week. The first step is to load the most recent raw data for all monitored customers from the respective data sources and conduct the necessary preprocessing steps, including feature generation. Afterward, all available and labeled samples are used to train a new model up to this time point. Once a new model has been trained, it is used to predict the customer sentiment of all monitored customers. Additionally, the SHAP values for each prediction and each feature are calculated. Predictions and SHAP values are then copied to a database and automatically loaded into an interactive dashboard which serves as a user interface.

- **Step 2: Identifying high-risk customers** One element of the user interface is an interactive table showing the most recent predictions for all monitored customers, along with some additional information about each customer, e.g., location and operating system type. Within this step, the user identifies a system within their area of responsibility with a particularly high probability for an escalation within the following two weeks. Once a customer has been identified, it can be selected, which reduces the information shown on the user interface to just the relevant information about the customer in question.

- **Step 3: Single out the most relevant features with SHAP values** Knowing only which customers are at high risk of causing escalations without knowing why is only of limited practical use. To explain why a specific customer has a high probability of escalation (customer sentiment), SHAP values for each prediction and feature are displayed in the user interface. With such a visualization, the user can easily single out one or a few features that greatly affect the customer sentiment according to their respective SHAP values. The information shown on the interface is further reduced to only display information connected to the selected customer and the selected features.

- **Step 4: Analyzing specific features** Once a set of features has been selected, the user is provided with the actual values of these features and how these values have evolved over the past weeks. With this information, the user can, for example, easily identify open yet unresolved tickets and see immediately for how long a specific ticket has been unresolved. Another example could be the accumulation of specific malfunctions reflected in log features.

- **Step 5: In-depth analysis of certain problems** At this point, the experienced user probably has a good idea of where a potential problem with the customer in question might be found (e.g., unresolved tickets, spare parts, or system malfunction). For an in-depth analysis of, for example, ticket data or consumed spare parts, other available tools are tailored for such tasks. Therefore, once the user has identified the potential root cause for bad customer sentiment, they are provided with a direct link to these external tools to continue the analysis as efficiently as possible to act before an escalation occurs.

The main idea of productive use of a data-driven decision support system is to help customer support employees to decide which customers to focus on and where to look.
VII. CONCLUSION AND FUTURE WORK

This paper proposes a general framework and an interactive workflow for a decision support system to monitor the customer sentiment for a large fleet of systems based on IoT (log) and enterprise data. Additionally, we provide a publicly available industrial benchmark dataset and all source code necessary to reproduce or improve our results. Our designed and implemented decision support system is currently deployed to monitor the customer sentiment of thousands of customers of high-end medical devices worldwide. The explainability of the system helps a variety of end-users to identify problems in the field. We demonstrate that using both log and enterprise data-based features enables more effective troubleshooting than using these data sources alone. Furthermore, the gained insights can help achieve better and more proactive customer relations and improve product management by focusing on problems that affect the customer the most. Our experiments show that with the proposed model M2 (RF with early fusion), 5 end-users who scan 10 customers per week could have prevented 48.61% of escalations within a year (Recall@50 = 48.61). Our experimental results serve as a baseline for the proposed problem. Additionally, we proposed a practical workflow for our designed decision support system framework.

Some open challenges could be addressed in future research using the provided benchmark dataset and evaluation framework. For example, more efficient methods for merging log and enterprise data information that preserve explainability can be investigated. Another challenge is to design models that, within the implemented framework, increase its predictive power without trading interpretability. Finally, alternative learning problem formulations, like anomaly detection, could be explored for the task. This could help with the heavy class imbalance present in the benchmark dataset.

APPENDIX

HYPERPARAMETER

Table 3 summarizes the hyperparameter search space for the classifiers used in our experimental study. The sampling strategies are according to the Optuna library [32]. The parameter names correspond to the respective implementation of the classifiers.

ACKNOWLEDGMENT

The authors would like to thank Gilles Le Texier, Martin Rothgaengel, Birgi Tamersoy, Mirko Appel, and Marie Mecking from Siemens Healthineers for their valuable inputs and discussions. The authors gratefully acknowledge Siemens Healthineers for providing the data. The authors would also like to thank Erick Axxe from The Ohio State University for proofreading our manuscript as a native speaker.

REFERENCES

[1] B. Hochstein, D. Rangarajan, N. Mehta, and D. Kocher, “An industry/academic perspective on customer success management,” J. Service Res., vol. 23, no. 1, pp. 3–7, 2020.
[2] Y. Lei, N. Li, L. Guo, N. Li, T. Yan, and J. Lin, “Machinery health prognostics: A systematic review from data acquisition to RUL prediction,” Mech. Syst. Signal Process., vol. 104, pp. 799–834, May 2018.
[3] Y. Lei, B. Yang, X. Jiang, F. Jia, N. Li, and A. K. Nandi, “Applications of machine learning to machine fault diagnosis: A review and roadmap,” Mech. Syst. Signal Process., vol. 138, Apr. 2020, Art. no. 106587.
[4] R. Sipos, D. Dradkin, F. Moerchen, and Z. Wang, “Log-based predictive maintenance,” in Proc. 20th Int. Conf. Knowl. Discovery Data Mining (ACM SIGKDD), Aug. 2014, pp. 1867–1876.

[5] M. Calabrese, M. Cimmino, F. Fiume, M. Manfrin, L. Romeo, S. Ceccacci, M. Paolanti, G. Toscano, G. Ciandrisi, A. Carrotta, M. Mengoni, E. Frontoni, and D. Kapetis, “SOPHIA: An event-based IoT and machine learning architecture for predictive maintenance in industry 4.0,” Information, vol. 11, no. 4, pp. 202, Apr. 2020.

[6] H. Liu and B. Lang, “Machine learning and deep learning methods for intrusion detection systems: A survey,” Appl. Sci., vol. 9, no. 20, p. 4396, Oct. 2019.

[7] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, “Deep learning for unsupervised insider threat detection in structured cybersecurity data streams,” in Proc. Workshops 31st AAAI Conf. Artif. Intell., Mar. 2017, Art. no. 00172.

[8] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, “Long short term memory recurrent neural network classifier for intrusion detection,” in Proc. Int. Conf. Platform Technol. Service (PlatCon), Feb. 2016, pp. 1–5.

[9] M. Landauer, F. Skopik, M. Wurzenberger, and A. Rauber, “System log clustering approaches for cyber security applications: A survey,” Comput. Standard and Interfaces, vol. 92, May 2020, Art. no. 101739.

[10] M. Du, F. Li, G. Zheng, and V. Srikumar, “DeepLog: Anomaly detection and diagnosis from system logs through deep learning,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., Oct. 2017, pp. 1285–1298.

[11] K. Zhang, J. Xu, M. R. Min, G. Jiang, K. Pelechrinis, and H. Zhang, “Auto-mated IT system failure prediction: A deep learning approach,” in Proc. IEEE Int. Conf. Big Data (IEEE BigData), Dec. 2016, pp. 1291–1300.

[12] Y. Baashar, H. Alhussian, A. Patel, G. Alkawsi, A. I. Alzahrani, O. Alifarrag, and G. Hayder, “Customer relationship management system (CRMS) in the healthcare environment: A systematic literature review,” Comput. Standards Interfaces, vol. 71, Aug. 2020, Art. no. 103442.

[13] Z. Soltani and N. J. Navimipour, “Customer relationship management mechanisms: A systematic review of the state of the art literature and recommendations for future research,” Comput. Hum. Behav., vol. 61, pp. 667–688, Aug. 2016.

[14] L. Montgomery and D. Damian, “What do support analysts know about their customers? On the study and prediction of support ticket escalations in large software organizations,” in Proc. IEEE 25th Int. Requirements Eng. Conf. (RE), Sep. 2017, pp. 362–371.

[15] C. Werner, Z. S. Li, and D. Damian, “Can a machine learn through customer sentiment?: A cost-aware approach to predict support ticket escalations,” IEEE Softw., vol. 36, no. 5, pp. 38–45, Sep. 2019.

[16] A. Nguyen, S. Foerstel, T. Kittler, A. Kurzyukov, L. Schwinn, D. Zanca, T. Hipp, S. Da Jun, M. Schrapp, E. Rothgang, and B. Eskofier, “Industrial benchmark dataset for customer escalation prediction, version 1.0,” Data Set, Zenodo, Dec. 2020, doi: 10.5281/zenodo.4383145.

[17] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee, “From local explanations to global understanding with explainable ai for trees,” Nature Mach. Intell., vol. 2, no. 1, pp. 56–67, Jan. 2020.

[18] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in Advances in Neural Information Processing Systems, vol. 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. New York, NY, USA: Curran Associates, 2017, pp. 4765–4774.

[19] R. Genuer, J.-M. Poggi, and C. Tuleau-Malot, “Variable selection using random forests,” Pattern Recognit. Lett., vol. 31, no. 14, pp. 2225–2236, Oct. 2010.

[20] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning (Springer Series in Statistics). New York, NY, USA: Springer, 2001.

[21] L. Breiman, “Random forests,” Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.

[22] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jun. 2016, pp. 785–794.

[23] J. H. Friedman, “Greedy function approximation: A gradient boosting machine,” Ann. Statist., vol. 29, no. 5, pp. 1189–1232, Oct. 2001.

[24] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic minority over-sampling technique,” J. Artif. Intell. Res., vol. 16, no. 1, pp. 321–357, Jun. 2002.

[25] H. He and E. A. Garcia, “Learning from imbalanced data,” IEEE Trans. Knowl. Data Eng., vol. 21, no. 9, pp. 1265–1284, Sep. 2009.

[26] G. Lemaitre, F. Nogueira, and C. K. Arid, “Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning,” J. Mach. Learn. Res., vol. 18, no. 17, pp. 1–5, 2017.

[27] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.

[28] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” IEEE Trans. Neural Netw., vol. 5, no. 2, pp. 157–166, Mar. 1994.

[29] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,” IEEE Trans. Signal Process., vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” J. Mach. Learn. Res., vol. 15, no. 1, pp. 1929–1958, 2014.

[31] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kêgl, “Algorithms for hyperparameter optimization,” in Proc. 24th Int. Conf. Neural Inf. Process. Syst. (NIPS). Red Hook, NY, USA: Curran Associates, 2011, pp. 2546–2554.

[32] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in Proc. 25th Int. Conf. Knowl. Discovery Data Mining (ACM SIGKDD), Jul. 2019, pp. 2623–2631.

AN NGUYEN received the B.Sc. and M.Sc. degrees in electrical engineering from the Technical University of Berlin, in 2016 and 2018, respectively, and the M.S.E. degree in electrical and computer engineering from the University of Michigan, in 2017. He is currently pursuing the Ph.D. degree with the Machine Learning and Data Analytics Laboratory, Friedrich-Alexander-University Erlangen-Nürnberg (FAU). His research interests include data science, machine learning, and process mining to analyze time series in industrial and healthcare applications.

STEFAN FOERSTEL received the B.Sc. and M.Sc. degrees in mathematics from Friedrich-Alexander University Erlangen-Nürnberg (FAU), in 2013 and 2015, respectively. He is currently pursuing the joint Ph.D. degree with OTH Amberg-Weiden under Prof. Dr. Eva Rothgang and the Machine Learning and Data Analytics Laboratory, FAU. Since 2015, he has been working for inforteam Software AG and currently heads the Center of Competence for data science and analytics topics.

THOMAS KITTLER received the B.Sc. and M.Sc. degrees in physics and the Ph.D. degree in astroparticle physics from Friedrich-Alexander-University Erlangen-Nürnberg (FAU), in 2012, 2015, and 2020, respectively. Currently, he is working as a Data Scientist with inforteam Software AG.

ANDREY KURZYUKOV received the B.Sc. degree in bioengineering system and technologies from Ural Federal University, in 2017. He is currently pursuing the M.Sc. degree in computational engineering with Friedrich-Alexander-University Erlangen-Nürnberg. He is working as a Business Data Analytics (BDA) at Siemens Healthineers. His research interests include data science, machine learning, and deep learning of time series.
Leo Schwinn received the B.Sc. and M.Sc. degrees in medical engineering from Friedrich-Alexander-University Erlangen-Nürnberg (FAU), in 2017 and 2019, respectively. He is currently pursuing the Ph.D. degree with the Machine Learning and Data Analytics Laboratory, FAU. His research interests include machine learning, deep learning, and adversarial machine learning to train robust neural networks.

Dario Zanca received the B.Sc. and M.Sc. degrees in mathematics from the University of Palermo, Italy, in 2013 and 2015, respectively, and the Ph.D. degree in smart computing from the University of Florence, Italy, in 2019. During his Ph.D. studies, he was partly working as a Visiting Researcher at the California Institute of Technology, California City. In addition, he served as a Postdoctoral Fellow at the Department of Medicine, Surgery, and Neuroscience, University of Siena, Italy. He is currently a Postdoctoral Fellow at the Machine Learning and Data Analytics Laboratory, Friedrich-Alexander-University Erlangen-Nürnberg, Germany. His research interests include in the areas of deep learning, computer vision, and neuroscience, with an emphasis on building artificial models that approach human capabilities of robust perception and reasoning.

Tobias Hipp received the Diploma degree in computer science and the Ph.D. degree in cryptography from the University of Ulm, in 2005 and 2008, respectively. Currently, he is working as a Data Scientist at Siemens Healthineers—customer service.

Sun Da Jun received the master’s degree in control science and control engineering from Shanghai Jiao Tong University, in 2012. Currently, he is working at Siemens Shanghai Medical Equipment Ltd., as a Big Data Software Engineer.

Michael Schrapp received the Diploma and Ph.D. degrees in general physics from Technische Universität München, in 2011 and 2015, respectively, and the master’s degree in stochastic engineering in business and finance from Hochschule München, in 2015. In 2015, he joined Siemens Corporate Technology as the Research Project Manager and later became a Senior Key Expert for industrial computed tomography. In 2017, he moved to Siemens Healthineers as the Head of Business Data Analytics, focusing on internal and field data analysis, as well as business intelligence. Since December 2020, he leads a program to enable customer sellable real-time monitoring and predictive services.

Eva Rothgang received the joint degree in informatics, with a focus on medicine, from the University of Ulm, Germany, and the University of Essex, U.K. During her Ph.D. studies, she visited the Center for Applied Medical Imaging (Siemens Corporate Research), which is part of Johns Hopkins Hospital, Baltimore, MD, USA. Her Ph.D. degree was supervised by Prof. Dr.-Ing. Joachim Hornegger (Pattern Recognition Chair, Friedrich-Alexander-University Erlangen-Nürnberg) and Prof. Dr. Gabor Fichtinger (Cancer Care Ontario Research Chair, Queen’s University, Canada). After receiving her Ph.D. degree, she held different positions at Siemens in the field of medical technology. Her journey at Siemens culminated in her being the Head of the Center of Competence for the topic of workflows with the Department of Product Definition and Innovation, Siemens Healthineers AG. In addition to profound knowledge in the areas of pattern recognition, artificial intelligence, and product development for medical technology, she has a deep understanding of not only the German but also the international healthcare industry. Furthermore, she holds 13 patents. Since May 2017, she holds the ZD.B Research Professorship in the subject area of digital process chains in medical care and technology at OTH Amberg-Weiden.

Bioern Eskofier (Senior Member, IEEE) received the degree in electrical engineering from FAU, in 2006, and the Ph.D. degree in biomechanics, for his research on application of pattern recognition methods in biomechanics under the supervision of Prof. Dr. Benno Nigg, from the University of Calgary, Canada, in 2010. In 2016, he was a Visiting Professor at Dr. Paolo Bonato’s Motion Analysis Laboratory at Harvard Medical School from February to March. In 2018, he was a Visiting Professor with Dr. Alex “Sandy” Pentland’s Human Dynamics Group at MIT Media Laboratory from March to August. He is also a Delegate of FAU to the Medical Valley (80 Mio Euro German Ministry of Education funded cluster) and to the European Institute of Innovation & Technology for Health (EIT Health, 500 Mio Euro EU Consortium, from 2015 to 2021). He has defined his research and entrepreneurial agenda to revolve around contributions to a digital health ecosystem, where patients are connected to other stakeholders within the healthcare systems using digital support tools. His digital health research philosophy is that only multidisciplinary teams of engineers, medical experts, industry representatives, and entrepreneurs will have the tools to actually implement changes in healthcare. He is a German Research Foundation (DFG) funded Heisenberg-Professor for digital support systems in sports and medical engineering and endowed Professor of Adidas AG. He heads the Machine Learning and Data Analytics (MaD) Laboratory and the Central Institute for Medical Engineering at Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU). Currently, his laboratory has 30 co-workers who research in the fields of machine learning and signal analysis for wearable computing systems in sports and health care. The motivation of the laboratory’s researchers is to increase human wellbeing. He authored more than 160 peer-reviewed articles, submitted five patent applications, and started three spinoff startup companies. He has won several medical-technical research awards.

***