# TECHNICAL UNIVERSITY OF MUNICH

## TUM Data Innovation Lab

## Interpretable AI for business applications


| | |
|---|---|
| Authors | Tuna Acisu, Soh Yee Lee, Almut Scheerer |
| Mentors | M.Sc. Olena Schüssler, Dr. Inna Vasylchuk |
| | Steering Lab by Horváth & Partners GmbH |
| Co-Mentor | M.Sc. Michael Rauchensteiner |
| Project Lead | Dr. Ricardo Acevedo Cabra (Department of Mathematics) |
| Supervisor | Prof. Dr. Massimo Fornasier (Department of Mathematics) |


Feb 2020

# Abstract

Artificial intelligence methods are revolutionizing many areas of business by providing better insights from data and more accurate predictions than ever. Especially in the field of predictive maintenance this is very useful as it presents the opportunity to foresee possible machine failures and minimize downtimes. But since many of those methods in AI depict highly nonlinear correlations and function as "black box" models, where the influence of the input features on the output is not transparent, the interpretability of those methods is very low. This is a big issue, especially for critical business applications, where the trust in the AI model has to be very high for it to be a useful tool in practice.

This challenge has given rise to the relatively recent field of eXplainable Artificial Intelligence (XAI), which describes multiple methods that can be used to improve the explainability of black box models. Our goal was to survey different XAI methodologies and compare their performance in the context of business applications.

After we implemented two different black box models for a predictive maintenance task, we transferred our methodology to the newly available client dataset. We achieved high accuracy in predicting events leading to downtimes of a compressor based on temperature and pressure data taken from the machine, using a deliberately opaque black box model of a neural network. Building upon this neural-network model we implemented two different XAI methods, namely LIME and SHAP, compared their outputs and verified our results with them. We succeeded at delivering a proof of concept model for the client, which can be practically applied to understand the factors leading up to different events and minimizing downtimes.

# Contents

# 1 Introduction

## 1.1 Problem Definition and Goals of the Project

With the enormous advancements in the area of computational power in recent years, more and more so-called "black box" models, e.g. Deep Neural Networks, are used for data analysis. To increase trust in these black box models and to be able to efficiently use them in business applications, explainable approaches are needed.

The goal of this project is to deliver a black box model and compare humanly interpretable representations of this model. To this end, a model dataset from the area of Predictive Maintenance is used and analyzed with a black box model to test the efficiency and explainability of different explainable AI models. Two different datasets were analyzed, one publicly available, the other one from a client of our project partner. The feasibility of our model together with the explainable AI methods will be used as a proof for potential clients in predictive maintenance domains.

In the following two sections, an introduction to Explainable Artificial Intelligence (XAI) is given as well as a short description of the field of Predictive Maintenance. Chapter 2 contains the analysis and different models for a publicly available battery dataset for Predictive Maintenance. The description, data handling and modeling of a real-world dataset available in the second half of this project can be found in chapter 3. Explainable AI methods for this dataset are described and compared in 4.

## 1.2 Explainable Artificial Intelligence (XAI)

There are many different machine learning algorithms which are used for Data Analysis. Every algorithm needs to be chosen carefully according to the available dataset as well as the target variable. The degree of interpretability of the model itself might also be a decision factor. Several models are potentially interpretable by design like decision trees, decision rules and linear regression. The opposite of that are black box models which cannot be understood by looking at their parameters or structure (e.g. a neural network), see [10].

Interpretability describes the degree to which a human can comprehend the decisions or output of a model [3]. But why is an interpretable model necessary? First of all, deep neural networks have proven to perform well learning highly nonlinear dependencies of possibly high-dimensional data [6]. Due to their better performance than interpretable methods in a lot of cases, black box models are needed as soon as accuracy of a prediction is crucial. In autonomous cars for example failure is not an option since it may lead to an accident. Understanding why the model predicted a certain output can help to learn more about the problem, the data and the reason why a model might fail [10]. This is also useful to know in order to leverage good outcomes. In some cases, the prediction by the machine learning model might not be the complete goal of the task and further information is needed to proceed.

A lot of tasks in business also require controlling mechanisms which cannot be provided by black box models. This as well as detecting biases in the model can be employed by interpretability. To increase the social acceptance and trust in an algorithm in a company, employees need to be provided with an explanation for its output.

As of the 8 April 2019, there exists a Ethics Guidelines for Trustworthy Artificial Intelligence presented by the High-Level Expert Group on AI, which was appointed by the European Commission. The Guidelines put forward a human-centric approach on AI and list 7 key requirements that AI systems should meet in order to be trustworthy [20]. This shows that the topic of accountability and interpretability is central and necessary to realize.

## 1.3 Predictive Maintenance

With the progress in accuracy of artificial intelligence methods, there has been the desire to apply those methods in an industrial setting. Predictive maintenance is such an application, where data of the state of a machine or some piece of equipment is collected, analyzed and used to predict the reason, location and time for necessary maintenance or failures in the future [14].

Usually this is done by collecting large amounts of data continuously and then applying machine learning methods, such as random forests or neural networks, to predict the state of the machine in the future. A typical target value in predictive maintenance is the Remaining Useful Life (RUL) [9], which tells the user how much longer the machine can operate productively. Alternatively one can predict whether the piece of equipment is working normally or whether there are any anomalies which need to be addressed.

Especially in predictive maintenance it is important for a machine learning model to be interpretable so that the user can comprehend why and how the model arrives at its conclusions. Maintaining or replacing a machine can be costly and require a lot of effort, so the owners should understand the recommendations of the algorithm and trust them enough to follow them. Conversely, by knowing which features the algorithm takes into account when predicting the RUL, the critical points can be identified and measures can be adopted to prolong the useful life of any piece of equipment.

# 2 Publicly Available Dataset

## 2.1 Dataset Selection

The first task in this project was to choose a suitable dataset for building a black box model. Since we did not know yet whether we would get access to actual customer data, we examined different publicly available datasets in the field of predictive maintenance.

In total we explored 12 different datasets and found 4 promising candidates: One dataset about coffee machine maintenance data [13], one with hard drive failure data [1], one with water pump maintenance data [19] and a dataset about battery life cycle degradation [16].

After some consideration we chose the battery dataset, since it has clearly labeled, interpretable features, almost no measuring errors or missing data points and a reasonable target variable to use for predictive maintenance (namely the Remaining Useful Life of the batteries). It consists of data taken from 124 batteries during 96.000 charging cycles in total, which was more than enough data to build a black box model.

## 2.2 124 Lithium-ion Batteries Dataset

The dataset of 124 lithium-ion batteries [15], as surveyed in [16], consists of 3 batches of battery charging data, each containing roughly 41 battery cells. The batteries in the dataset were cycled to failure under fast-charging conditions with a different charging policy per batch. The policies where two-step charging policies, where the (constant) current changes after a certain state-of-charge. The batteries were cycled "to failure", meaning until the battery does not charge to 80% of full capacity anymore. The "cycle life" of the battery i.e. the number of cycles till failure, are the target feature we try to predict.

The data provided in the set is tripartite: Beyond descriptors for each battery (tag number, charging policy and cycle life), it contains summary features for each of the charging cycles, consisting of cycle number, discharge capacity, charge capacity, internal resistance, maximum temperature, average temperature, minimum temperature, and chargetime. In addition to that, the dataset also contains detailed cycle data for the entire duration of each charging cycle for each battery. This consists of the timestamp, charge capacity, current, voltage, temperature and discharge capacity. With all in all 96,700 cycles, this is the largest publicly available lithium-ion batteries dataset.

For the data to be suitable for our use case, the variability in batteries charged with the same policy had to be high enough, so that our black box model could learn other important features. Since this is the case and the data is reasonably clean, we had a good basis to work on our black box model.

**Train / Validation / Test split** The dataset split for training (41 cells), validation (43 cells), and testing (40 cells) was already available in the simple preprocessing code

given by the authors. We followed this split when building the models described in the next sections.

## 2.3 Linear Model as Benchmark

We started by building a white-box linear model explained in the journal paper [16]. The authors used their domain knowledge in lithium-ion batteries to engineer features with high predictive potential. Our idea was to apply XAI methods on Deep Learning Models and compare the results with this linear model.

**Feature Engineering** We implemented the features proposed by the authors in their regression model, for details see [17]. The key idea is that the battery capacity is considered as a function of voltage, as opposed to voltage as a function of capacity, in order to maintain a uniform basis for comparing cycles. The discharge voltage between cycle 100 and 10 is denoted $\Delta Q_{100-10}(V)$. Our linear model features were:

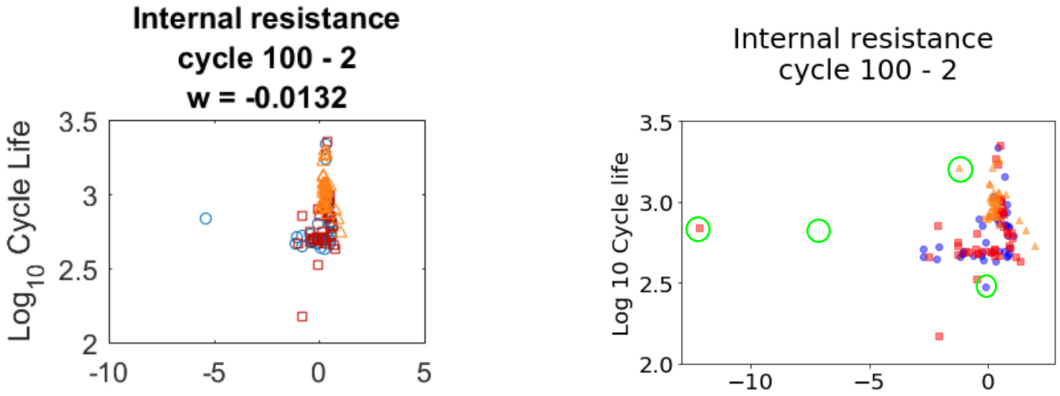
- Minimum $\Delta Q_{100-10}(V)$
- Variance $\Delta Q_{100-10}(V)$
- Slope of the linear fit to discharge capacity, from cycles 2 to 100
- Intercept of the linear fit to discharge capacity, from cycles 2 to 100
- Discharge capacity of cycle 2
- Average discharge time, from cycle 2 to 6
- Minimum internal resistance, between cycle 2 to 100
- Difference internal resistance, between cycle 100 and 2

**Modeling** Our target was to predict the logarithm of cycle life for batteries. We performed a Z-score standardization on the training set and applied it to the validation and test sets. Then, we built an elastic net model as advised by the authors because it could deal with high correlations between the features. The model was trained with a four-fold cross-validation on the training set, tuned with the help of the validation set and verified on the testing set.

**Challenges** To reproduce the published result, we plotted our features corresponding to cycle life and compared this to the feature plot by the authors, see figure 1. The graphs have a similar overall shape, but the authors might have done some numerical rounding and excluded outliers without mentioning the details in their publications, which would explain the discrepancies.

Besides that we also excluded the temperature integral from cycle 2 to cycle 100 from our model. This was done for two reasons: first, we still saw a big difference when comparing the authors' plot for this feature with ours and second, a much worse result was returned

when including this feature. A similar question was found in their GitHub [15] without useful answer. We finally requested access to the authors' modeling code but did not receive any response when moving on to the real-world dataset.




(a) A feature plot by authors

(b) Our feature plot

Figure 1: Feature internal resistance difference between cycle 100 and cycle 2 corresponds to battery cycle life. The train, validation, and test sets are represented by blue circles, red squares, and orange triangles, respectively. The discrepancy is circled as green in 1b.

**Result**

| Training set | | Validation set | | Test set | |
|---|---|---|---|---|---|
| MAE (cycle) | MPE (%) | MAE (cycle) | MPE (%) | MAE (cycle) | MPE (%) |
| 71 | 11.68 | 126 | 29.18 | 164 | 15.03 |

We had a high error in the validation set due to outliers that can be seen in figure 1, which might be due to a different preprocessing for outliers of the authors without including the details in their publication. Nevertheless, this result is ready as a benchmark model.

## 2.4 Deep Learning Models

In order to implement and compare various XAI methods, a black box model had to be implemented first. We approached two different Neural Network architectures which are described in the next two sections. After implementing those models, we got the exciting news to work on a real-world dataset for our supervisor's client. Thus, we stopped working on the battery dataset and proceeded with the new challenge. Our approach and results are described in the next chapter 3.

### 2.4.1 Convolutional Neural Network (CNN)

After researching various papers on Neural Networks built for battery datasets to get an idea about possible architectures, we found a blog and GitHub repository that implemented a CNN in Python on the 124 batteries dataset ([5], [4]). Thus we proceeded by using the code provided in the GitHub repository as a foundation for our own architecture.

**Modeling** The data analysis and CNN provided in the GitHub repository [4] is based on the dataset [15]. The goal of the CNN model was to build a more sophisticated model and to be able to feed the model measurement data from a battery that is already in use.

To achieve this goal, they only used the data from the discharging periods since the charging policy varies from cell to cell and therefore the charging measurements are not comparable, whereas discharging measurements are. Batteries that did not reach the 80% capacity were excluded. Additionally, cycles with time gaps, small outliers, or other inconsistencies were removed. For smoothing they used the Savitzky-Golay filter. Further, the data for charge, voltage and temperature was not comparable across cycles and batteries since the different charging policies lead to different cycle time frames. Thus they took the voltage range instead of time as reference and interpolated charge and temperature over 1000 equidistant voltage steps. Then the detail features per cycle included into the model are charge (Ah) and temperature (°C) and the scalar features per cycle included into the model are Internal Resistance (Ω), Total Charge (Ah) and Discharge Time (minutes).

The model is fed the features of 20 consecutive cycles of one battery. Its architecture is shown it figure 2. The detail features are processed in three Conv2D layers with MaxPooling, the scalar features are processed in two Conv1D layers with MaxPooling. The results are flattened, concatenated and sent through a fully connected dense network. The two outputs are the current cycle (showing the current age of the battery) and the remaining cycles (providing the Remaining Useful Life). The total cycle life of one battery can then be computed by summing up the two outputs. This is comparable to the output of the linear model being the logarithm of the total cycle life.

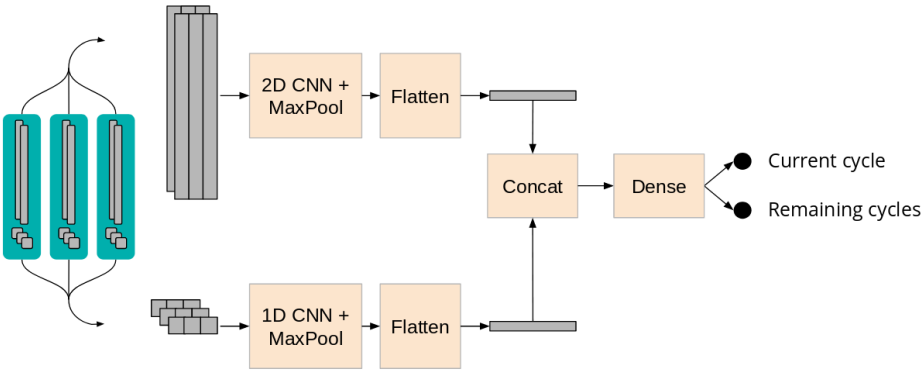



Figure 2: Architecture of the CNN model implemented in [4]. Picture taken from [5].

**Result** After resolving several issues with the provided code and fixing the saving and loading logic of the model that arises from tensorflow library, our results are in a similar range than the results achieved by the authors of the blog [5]. The metrics used are loss and mean absolute error (MAE) of the current cycle, as well as for remaining cycles. The MAE is calculated as difference between the output and the real value for each battery, then averaged and multiplied with the maximum battery cycle lifetime. The best result of the authors of the code in [5] was 90 MAE for the current cycle and 115 MAE for the remaining cycles of the validation dataset. Our best result after 250 training epochs was:

| Training set | | | Validation set | | |
|---|---|---|---|---|---|
| loss | MAE current cycle | MAE remaining cycles | loss | MAE current cycle | MAE remaining cycles |
| 0.0050 | 92.6132 | 104.9049 | 0.0079 | 91.6973 | 132.3842 |

**Additional Features** Since XAI methods are mainly a visualization of feature importance or explain results based on certain values of features, more features were desirable to perform XAI analysis on this model. We thus added an alternation of the features that are used in the linear model described in section 2.3. These features are window-based, i.e. on 20 consecutive cycles, compared to features based on cycle 2 to 100 or 2 to 6 in the linear model. The detailed added features can be found in the appendix (table 4). These window-based features were inserted into the dense layer of the CNN model.

**Result With Additional Features** The best result in 500 training epochs was:

| Training set | | | Validation set | | |
|---|---|---|---|---|---|
| loss | MAE current cycle | MAE remaining cycles | loss | MAE current cycle | MAE remaining cycles |
| 0.0070 | 123.1916 | 117.9473 | 0.0079 | 123.3576 | 116.0443 |

This result is slightly worse than the result without additional window-based features. Possibly a change in the CNN architecture could resolve this. Since we then got the real-world dataset, we did not improve the model any further.

### 2.4.2 Recurrent Neural Network (RNN)

Since the battery dataset consists of time-series for each charging cycle as well as one time-series of summary data per battery, we decided to follow an alternative approach with an architecture often used for time-series data: A Recurrent Neural Network.

For the RNN we used a basic model having one LSTM layer with 128 nodes, dropout regularization (30%) and two dense layers. The input was the summary data Internal Resistence, Total Charge and Discharge Time, taken over a sliding window of 20 cycles somewhere in the lifecycle of the battery and engineered in the same way as in the CNN (see also table 5). Detailed cycle data was not included and the two outputs are again the current cycle and the remaining cycles.

**Result using Summary Features** The best result in 400 training epochs was:

| Training set | | | Validation set | | |
|---|---|---|---|---|---|
| loss | MAE current cycle | MAE remaining cycles | loss | MAE current cycle | MAE remaining cycles |
| 0.0086 | 138.6296 | 120.0887 | 0.0166 | 164.0829 | 162.7043 |

As can be seen from the table, the results of the RNN are a bit worse than the CNN results, but since the RNN is only using the one-dimensional summary features, this was to be expected. With further tuning this could be improved, but since we then got access to the real-world dataset, we stopped the development of the RNN.

# 3 Industrial Hydrogen Compressor Dataset

## 3.1 Introduction to the Compressor Dataset

At the end of November 2019, a real-world dataset from a client of the Steering Lab by Horváth & Partners GmbH was made available to us.

This dataset contains sensor data taken in short, irregular steps from both turbine and compressor in a site. A sketch of the compressor can be found in figure 12 in the appendix. In total less than 100 different sensors measured the rotational frequency, pressure, temperature, thrust and re-circulation at different points of the machine. Not all sensors readings cover the whole time period as some were installed later. This was a challenge for the bearing damage analysis. Besides having the dataset, we also received repair dates and some report information on events and repairs.

We were asked to look at 4 possible events and predict indicators for these events:

- Valve tightening (routine maintenance or when weird sound is heard)
- Valve breakage (leads to machine stop)
- Bearing damage (machine stop)
- Change of oil (routine maintenance)

We discarded the change of oil as target variable since there were only three maintenance cases in the whole time period. The bearing damage events can be found in the documentation of Anes Valentic. Therefore, we are only looking at two events related to valves.

## 3.2 Data Sanity Check

From the report information on events and repairs we first had to extract the events together with our project partner. Some technical understanding of the machine was needed for that. This was additionally difficult since not all downtimes were linked to either valve breakage or bearing damage, but scheduled maintenance independent of the repair events.

Sensors only record a data point if it is deviating a certain amount from the last measurement. To make the measurements comparable across different sensors, the data was resampled and the measurements linearly interpolated.

Before we dove deep into the analysis of the data, we first conducted a sanity check by checking the downtimes and events we received from the factory against the rotation of the turbine. During bearing damage or valve breakage the machine should not be operated anymore and the rotations per minute of the turbine should go to zero. This was true for all events and allowed us to determine date and time the turbine stopped and started again additionally to the date specified in the report information.

We also performed a short consistency check: the data that was given to us in two formats, once clustered by year and once by sensor, was consistent. Some missing sensors that were given in a description but not as data was detected, but this did not affect our modeling process. We also cleaned duplicates records.

## 3.3 Exploratory Data Analysis for Valve Tightening Events

We analysed all valve tightening events and plotted all sensors in the compressor to find abnormal patterns, but no obvious pattern was detected. However, we discovered that some valves break soon after tightening, which may hint that the valves were fastened too loose or too tight. From the repair records, we also saw that more valve tightening was done after certain years and since then no valve breakage was reported anymore. We then assumed more valve tightening leads to less valve breakage which the client confirmed.

We also dove deep into more critical events in which valves were loose enough to hear obvious noise. Majority of these events had breakage patterns similar to figure 3.

As more serious valve loose events have similar abnormal patterns as the ones before valve breakage, we focused on building a model for valve breakage events.

## 3.4 Exploratory Data Analysis for Valve Breakage Events

Based on the information we received from the client and the domain information acquired over time, we decided to focus our analysis and model for the valve breakage on the information provided by the temperature sensors.

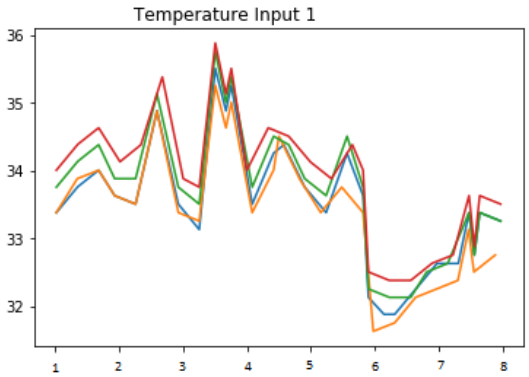
As can be seen in the diagram of the compressor (figure 12 in the appendix), there are temperature sensors on each valve. The sensors measuring the input flow generally have a much lower temperature. In addition to that there are also pressure sensors measuring the input and output pressure of cylinders 1 & 2 as well as 3 & 4 separately.

For the first exploratory data analysis of these temperature values we focused on the behavior shortly before and after the downtimes and tried to see if we could find any patterns regarding the temperature changes.

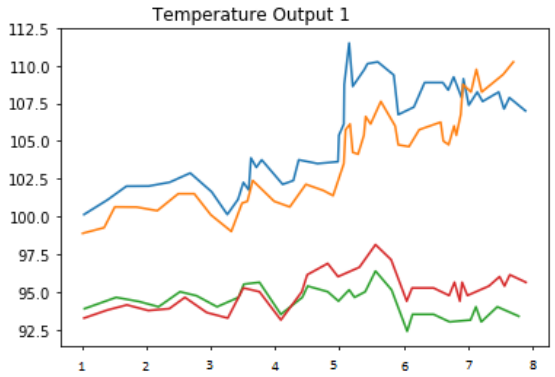
What we found gave us a first indication on how to predict an approaching downtime for the events of valve breakage. Shortly before the valve breakage event, the temperature difference between the input and output sensors goes up (i.e. the input is colder and the output is hotter), which makes sense, considering that the compression happening inside the cylinder cannot work properly if the valves are broken. Figure 3 shows one example for this.

## 3.5 Challenges in Modeling

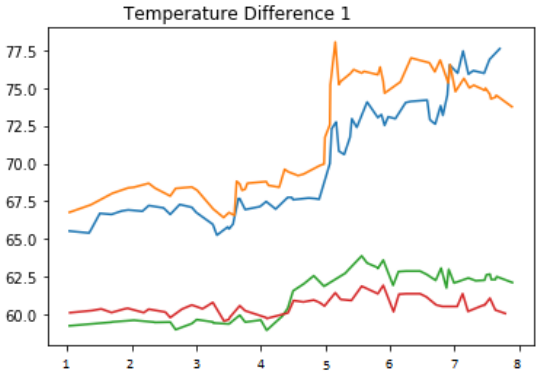
While working on completing the machine learning model and the implementation of the XAI methods in less than 1.5 months, we faced different challenges, mainly due to




(a) Input temperatures of cylinder 1




(b) Output temperatures of cylinder 1




(c) Differences in temperatures

Figure 3: Analysis of valve sensors

incomplete information and "messy" raw data:

- It is not possible to differentiate between real valve breakage event and a regular maintenance event. We did not know the exact starting time when a valve operates abnormally.
- The dataset is extremely imbalanced. There were less than 1% valve breakage events recorded.
- There are more than one valve in the compressor in total, but the report information on events and repairs does not contain which valves were broken.
- Special events that create irregular readings at the sensors:
  - The whole machine may be shut down for repair, upgrade and maintenance.
  - Some valves may be closed if the current speed of the turbine is insufficient to run the compressor in full capacity.
- Complex seasonality of temperature sensors. They are influenced by outside temperature that are composed of day and night season and summer and winter season.
- Valves only broke in the first few years, but the model should be able to predict this event far beyond the interval, until recent years. There were several reconstructions

and upgrades of the machine without additional details that caused a different data distribution in the later years.

Although it was very challenging to formulate the exact problem, we finalized our machine learning problem by predicting valve breakage *on a daily basis* of $n$ valves as a *whole*. It means that our model would learn and predict event class as **1** if any of the valves started to run abnormally until they broke down or a repair took place, and **0** if all valves are operating normally.

We chose a daily instead of weekly aggregation of the raw time-series as the abnormal pattern typically lasts for a few days. This, however, does not tell which valves are abnormal. We then applied XAI methods (see chapter 4) to identify problematic valves to simplify machine maintenance and provide further insight.

The solutions for the mentioned challenges are explained in the following sections.

## 3.6 Data Cleaning and Preprocessing

To create a proof of concept model that delivers high business impact for the client of Horváth & Partners GmbH, we spent a lot of time on data preprocessing to solve the challenges written in chapter 3.5 to produce high-quality data. The following section is divided into subparagraphs to discuss each challenge and solution in details.

### 3.6.1 Label Valve Breakage Events Manually

As mentioned previously, the raw data does not truly reflect the health state of valves. Those valves might already behave abnormally before detected by technicians. Therefore, we went through all breakages written on the repair records, plotted graphs for more than 30 sensors, and compared the graphs with the actual valve breakage date. We extended the event duration and set the event class to 1 as long as an abnormality could be seen in the plots.

For example, in figure 3, a valve breakage was reported on day 1, but we already saw the abnormal pattern in a plot starting day -5. Therefore, we set the breakage event to 1 starting day -5 until a repair took place on day 8. This step is tedious yet critical, as it enables our model to detect the abnormal pattern before valves break.

### 3.6.2 Handle Downtimes

Sometimes the whole machine would be shut down due to events, restarts, and occasionally hardware upgrades and not all of them were related to valve breakages. During shutdowns, sensors' readings drop to their "zero" states, which are significantly different from the regular values. These dramatic changes make model training much more difficult, and thus required special processing.

**Definition of a Shutdown** First, we compared three methods to extract all downtimes: either looking at the turbine output (rotation per minute = 0) or valves outputs (output

pressures = 0 for cylinder 1 & 2 and 3 & 4, or outputs temperature sensors of cylinder 1, 2, 3, and 4 drop below a certain threshold). We proceeded with the output temperature sensor of cylinder 2 because:

- Turbine output fluctuates significantly, and it could be just a quick restart
- There were spikes in the temperature readings due to repairs or valves closure without turbine shutdown
- The output pressure of the valves is rather constant, making it difficult to find a good threshold
- There are more breakages at cylinder 1 & 2 compared to 3 & 4, and the output temperature of cylinder 2 is more stable than of cylinder 1
- This selected downtime may be extended easily to include partial closure of valves

**Healthy Intervals Extraction** If the output temperature sensor of cylinder 2 falls below $t$°C, we consider it a *valid* shutdown.

To extract the "healthy" interval after the previous step, we calculated the average temperature drop per minute, 1.2, and its standard deviation, 0.6, for all shutdowns based on the output. A pessimistic temperature drop per minute, 0.305, could then be derived from these statistics. We concluded that it took less than 60 minutes to drop 20°C from normal operating temperature for 99.7% of cases. Besides, we got the important information from the client that it takes up to 2 days after a downtime for the machine to stabilize and operate normally again. Thus, we defined the healthy intervals as 2 hours before the output is less than $t$°C, and 2 days after the output is greater than or equal to $t$°C again, as summarized in figure 4.

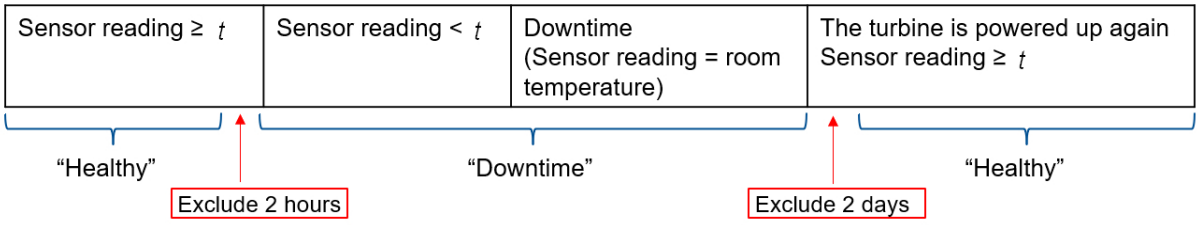



Figure 4: Extraction of healthy intervals among shutdowns

We exported our result to a .csv file to have more control and also eliminated a very short healthy interval (less than 7 days) between shutdowns.

**Application in the Model** There are two choices: (1) replace the values in shutdown intervals with interpolated values using the healthy intervals before and after the shutdown, or (2) just remove them entirely when training a model. As a shutdown happened after an interesting breakdown event, we chose (2) to prevent our model from learning the interpolated patterns that are not available when predicting events in the future.

### 3.6.3 Remove Influence of Outside Temperature on Temperature Sensors

Since the main input of the model are the temperature sensors located on the compressor valves, it is important that this input is as meaningful as possible. Influence from the outside temperature is therefore undesirable and should be minimized to isolate the pure change in temperature that is coming from the machine valves.

We first attempted to remove the influence by doing a seasonal decomposition of the outside temperature, decomposing the seasonality in daily and yearly cycles (for the day/night and summer/winter difference in temperature) and then downscaling the seasonal cycles and subtracting them from the temperature readings of the sensors.
The downscaling factor per valve temperature sensor is needed since the valves are less influenced by outside temperature changes than the actual outside temperature. To determine the right choice of this factor was a difficult task and the result of this first attempt still showed outside temperature influences.

As a final approach we used adaptive filtering to improve the input to our model. It allowed us to very accurately filter out the exact influence of the outside temperature, beyond just seasonal effects.

**Theory** The following explanation is taked mostly from [21]. Adaptive filtering is an approach from signal processing, where two input signals, $d_k$ and $x_k$, discrete measurements at different time points, are fed into the filter. Here $x_k$ is the *observation*, i.e. the actual signal that is measured, and $d_k$ is the *desired signal*, i.e. the signal we want to isolate from the observation.

In our case we have the raw valve temperature readings of the sensors as $x_k$ and the outside temperature as $d_k$. The signal we actually desire in this setting would then be the *residual* $\epsilon_k$, which is the signal which is left over once the influence of the desired signal is removed.

Adaptive filtering works by learning the weights that allow the reproduction of the desired signal from the observation, see also figure 5. The output from the filter is the reproduced desired signal $y_k$. It is computed by applying the weights to the input $x_k$:

$$y_k = \sum_{n=1}^{N} w_n * x_{k-n} \,.$$

The residual signal is then defined as

$$\varepsilon_k = d_k - y_k \,.$$

This is the signal which is left once the influence of the desired signal is removed. It gives the temperature changes in the machine, which are not explained by the outside temperature changes.

**Practical Application** To implement this filter, we used the Python library padasip [2] with a ready-built adaptive filter.

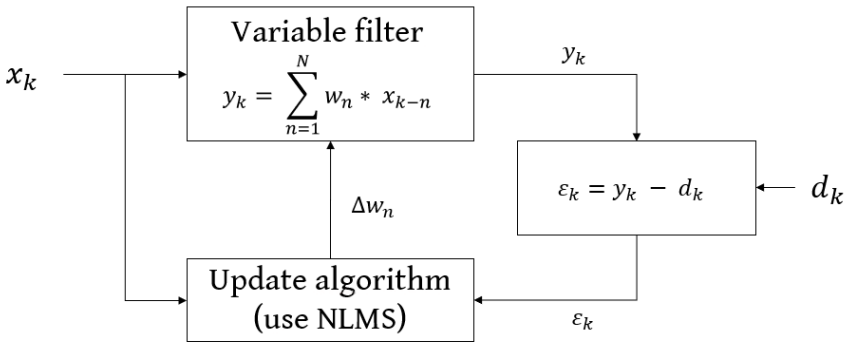



Figure 5: Adaptive Filtering

Before filtering we smoothed the outside temperature measurements, so the filter does not learn every little fluctuation, but only actual reasonable trends. For the smoothing we used a time frame of 4 hours and applied a Savitzky-Golay filter of polynomial order 3.

For the adaptive filtering we then used interpolated data on a minute basis. We chose the last 60 measurements of the outside temperature as the input to the adaptive filter ($N = 60$), so the adaptive filter learned the influence of the last hour of outside temperature on the valve temperatures. As error function we chose Normalized Least Mean Squares.

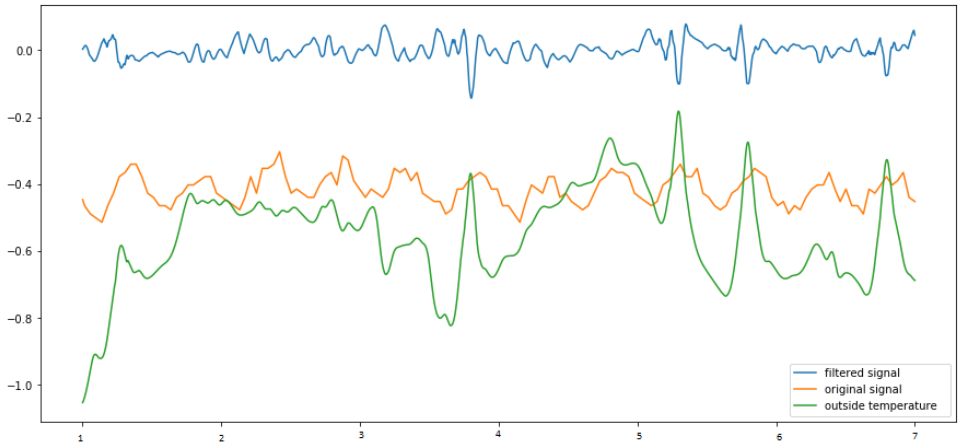
**Result** The result of the temperature removal are the cleaned temperature sensor readings. As can be seen in the figures, the correlation of the outside temperature and the residual signal is completely gone in 6a and one can very well see the changes in temperature that a downtime or an abnormal event brings in 6b. The same is depicted in table 1 showing the correlation of the outside temperature and valve temperature sensor T3041 (unfiltered and filtered) during normal operating conditions and table 2 for abnormal conditions.

| | outside temp | raw valve temp | filtered valve temp |
|---|---|---|---|
| outside temp | 1.00 | 0.47 | 0.01 |
| raw valve temp | 0.47 | 1.00 | 0.09 |
| filtered valve temp | 0.01 | 0.09 | 1.00 |

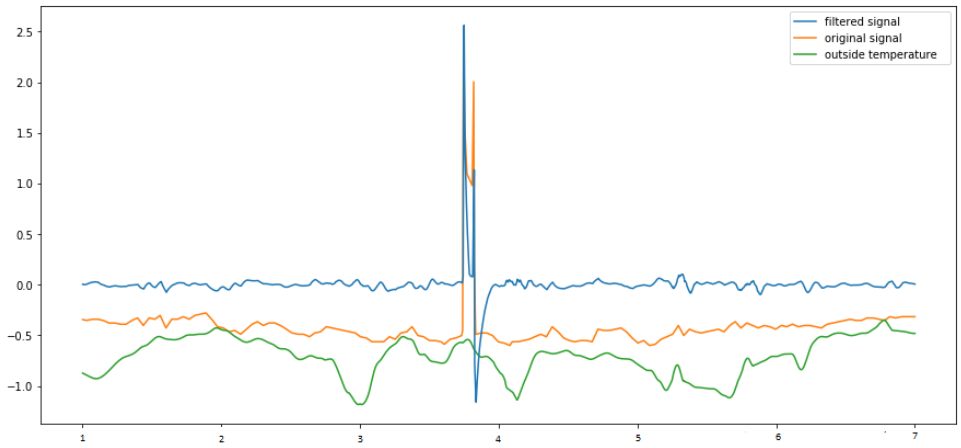
Table 1: Correlation under normal conditions

| | outside temp | raw valve temp | filtered valve temp |
|---|---|---|---|
| outside temp | 1.00 | 0.31 | 0.01 |
| raw valve temp | 0.31 | 1.00 | 0.58 |
| filtered valve temp | 0.01 | 0.58 | 1.00 |

Table 2: Correlation under abnormal conditions




(a) Raw temperature




(b) Cleaned temperature

Figure 6: Raw and cleaned temperature of a sensor.

## 3.7 Modeling

Our goal was to first build and then explain the output of a black box model with XAI methods. Thus, we implemented a multilayer perceptron (MLP) model that could easily capture interactions among features, yet is complex enough to apply XAI methods to meet the goal of this project.

The raw time-series data is not yet ready to feed into models. Figure 13 in the appendix summarizes the crucial steps before feeding inputs into a model.

More than 1000 models with different feature engineering, aggregation, and hyperparameters were built and evaluated using the CRISP-DM methodology. This chapter lists the details of the final models whose results were already presented to the client.

| | **Model (1)** | | | **Model (2)** | | |
|---|---|---|---|---|---|---|
| | $f$(valve 1 high temp - valve 1 low temp) | $f$(valve 2 high temp - valve 2 low temp) | ... | $f$(valve 1 high temp) | $f$(valve 2 high temp) | ... |
| day 1 | ... | ... | ... | ... | ... | ... |
| ... | ... | ... | ... | ... | ... | ... |
| day $n$ | ... | ... | ... | ... | ... | ... |

Table 3: Different features used in Model (1) and Model (2). $f$ is any function applied per day during aggregation. We let $f$ as $max$ in Model (1) and $min$ in Model (2)

### 3.7.1 Feature Selection and Engineering

From data exploration analysis (see section 3.4) and rapid prototyping, we identified that the output temperature sensors of each valve in the compressor (see figure 12) are crucial to identify breakages. According to the thermodynamic law, the gas temperature goes up during gas compression. When a valve is loose, it may be needed to add more pressure to compress the gas, which could result in a higher output temperature that indicates abnormally. The sensors are, however, influenced by outside temperature, and we had to remove this influence to get better prediction. The steps are described in 3.6.3.

While working on a function that removes the outside temperature influence, we also built another model with creative feature engineering that gave a much better result than the raw features. By pairing input and output temperature for each neighboring valve and taking the difference between them, for instance, $V1_{out} - V1_{in}$ in figure 12, we managed to create a better model that pushed the AUC score from 0.66 to 0.9.

Table 3 summarises the models delivered with this project.

We also applied Z-score standardization to the whole dataset based on the training set. Besides that, all numeric values are bounded to a reasonable minimal and maximal value, as extreme values do not give additional information for valve breakages but only make the model more difficult to train.

As we already obtained an excellent result with this creative feature engineering, we decided not to create more complex features since this may lead to difficult interpretation in the XAI phase.

### 3.7.2 Balancing Dataset

There are only less than 1% valve breakage events (positive class) in the raw time series, which makes the model very challenging to train. We applied several methods to dramatically improve the model performance:

1. Label valve breakage events manually
2. Cost-sensitive learning
3. “Manual” oversample minority class

We already discussed method 1 in 3.6.1.

Method 2 enables the model to put more weight on examples from an under-represented class. We did this by calculating the ratio of positive class versus negative class and passed it as a parameter when fitting the model.

As for method 3, we did not use the oversample logic from Python's sklearn library but rather manually inserted a few more data points of positive examples instead of using only one data point per day when doing daily features aggregation. By balancing the class distribution of the dataset with this method, we saw a much better model having a stable loss in both the training and validation phase.

### 3.7.3 Train / Validation / Test Split

We used a holdout approach guaranteeing that data points of an event belong to either the training, validation or testing set. On the other hand, in a first naive cross-validation approach, the data points of the same event may be found in all train, validation, and test sets. This naive approach caused overfitting and overconfidence in the model's performance.

We also ensured the training set contains data of all valves that failed before for the following reason: if an abnormally of e.g. valve 1 was only found in the testing set but not the training set, the model would perform badly because it has never seen this pattern during the training phase. We "hand-picked" holdout split for this project with the fraction: 45% train, 30% validation, 25% test. Two years were excluded due to major reconstruction work, long downtimes, and frequent machine restarts.

Another possible split is with a rolling window approach. We did not proceed with this approach because it requires more training time. In addition to that, the holdout approach gives more a accurate information on the model's performance.

**Limitation** As some valves never break before, our model will not be able to recognize breakage pattern on these valves.

### 3.7.4 Final Models

We kept our MLP architecture as simple as possible to prevent overfitting. The models takes the input features described in section 3.7.1. The output sigmoid layer returns the probability that the valves are operating normally on a given day:

| Layer | Number of nodes | Activation function | Dropout probability |
|---|---|---|---|
| Input layer | number of features | - | - |
| Hidden layer 1 | 50 | selu | 0.5 |
| Hidden layer 2 | 50 | relu | 0.5 |
| Output layer | 1 | sigmoid | - |

**Result** Figure 7 shows the best result after at most 100 epochs, with early stopping when the validation loss does not improve anymore after $n$ iterations: Model (1) uses the "pair" features; Model (2) uses the "cleaned" temperature of each valve as features.

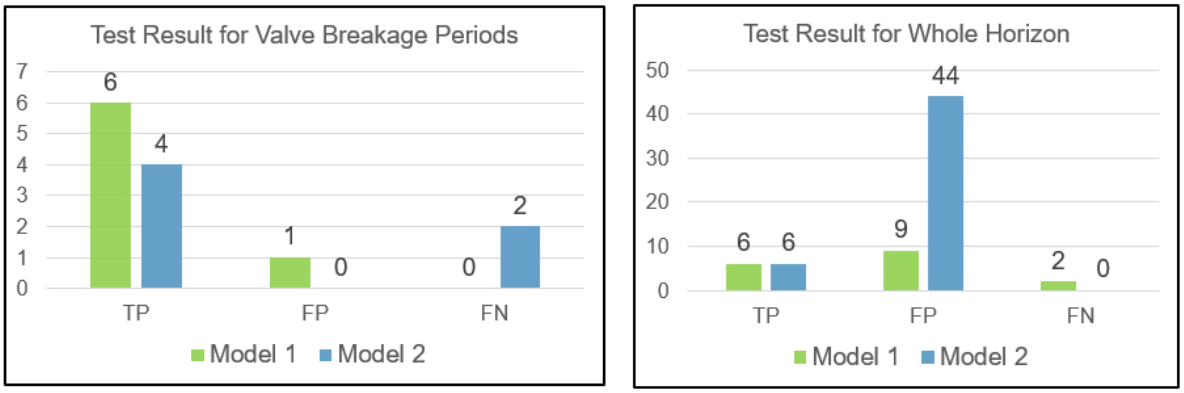



Figure 7: Results for Model (1) and Model (2). True Negative result is not shown in this plot.

We took a closer look into the cases where model (1) gives false positive for the whole horizon. More than 90% of all false-positive cases happen right before the event dates, and this is expected because we did not have an expert to verify our manual event labeling. This means the model is so good that it finds trivial abnormal patterns that are not visible with the bare eye. In other false-positive cases, there might be a spike on the sensor reading at a particular day that goes back to normal again in the next day and our model captures the spike too. To exclude this case, we recommend that the client only investigates further if the model predicts an abnormality for more than 2 consecutive days.

Model (2) has much more false alarms which could not be explained easily. It could be that the influence from higher room temperature may lead to more breakage, this information is removed when removing the outside temperature. We also tried to "pair" these clean readings by taking the difference of input and output temperature of each valve as in model (1), but the result is worse than the one of model (1).

Nevertheless, these results are sufficiently good to apply XAI methods to find abnormal sensors.

# 4 Explainable Artificial Intelligence (XAI)

As a final step of our project we implemented two XAI methods for the dataset, see chapter 3. Their theory and our results are described in the next two sections.

## 4.1 Local Interpretable Model-Agnostic Explanations (LIME)

LIME [12] explains the predictions of any classifier in an interpretable and faithful manner by learning an interpretable model locally around the prediction. Its greatest advantage is that it is independent of the machine learning model it explains.

**Theory** Let $G$ be a class of potentially interpretable methods (e.g. linear models, decision trees) and $g \in G$ an explanation model with complexity $\Omega(g)$ (e.g. number of non-zero weights for linear models, depth of the decision tree). Let $f$ be the model being explained and $\pi_x(z)$ be a proximity measure between an instance $z$ and $x$. $L$ is a measure of unfaithfulness of $g$ in the local approximation of f. The explanation is then given by:

$$\text{explanation}(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega(g).$$

This is usually achieved by fixing $\Omega(g)$ and approximating $L(f, g, \pi_x)$ as follows:
Sample around the instance of interest $x$ and get the black box predictions for these new points. Weight the new samples according to their proximity to $x$ and train a weighted, interpretable model on this dataset. This local model can then be interpreted.

**Python Library** In the Python library *lime* [11] a linear model is used for explanation. The sampling of the new data points is done with a Gaussian distribution. The complexity is determined by the maximum number of features the linear model may use. LIME uses an exponential smoothing kernel to define the neighborhood, where the kernel width is set to $0.75\sqrt{\#\text{ features}}$.

**Application in the Model** We applied LIME to model (1) described in 3.7, where no outside temperature removal was done and the prediction was restricted to cylinder 1 and 2. The model was trained on data containing valve breakage events only instead of whole horizon. The linear model of LIME receives the quantiles of the numeric features used in the machine learning model.

The result of LIME can be seen in figure 8. For each day, the left most part gives the prediction probabilities for class “Working” and “Abnormal”. The middle part gives the 8 most important features in descending order. Attributes having orange color support “Abnormal” and those with blue color support “Working”. The float point number on the horizontal bars represent the relative importance of these features and their weight in the linear model. For example on day 2, if “S1” had been small or equal $-0.18$, the probability for “Abnormal” would drop by around 39%. The right part for each day follows the same color coding as the other two parts. It contains the actual values of the features ranked by their importance given in the middle.

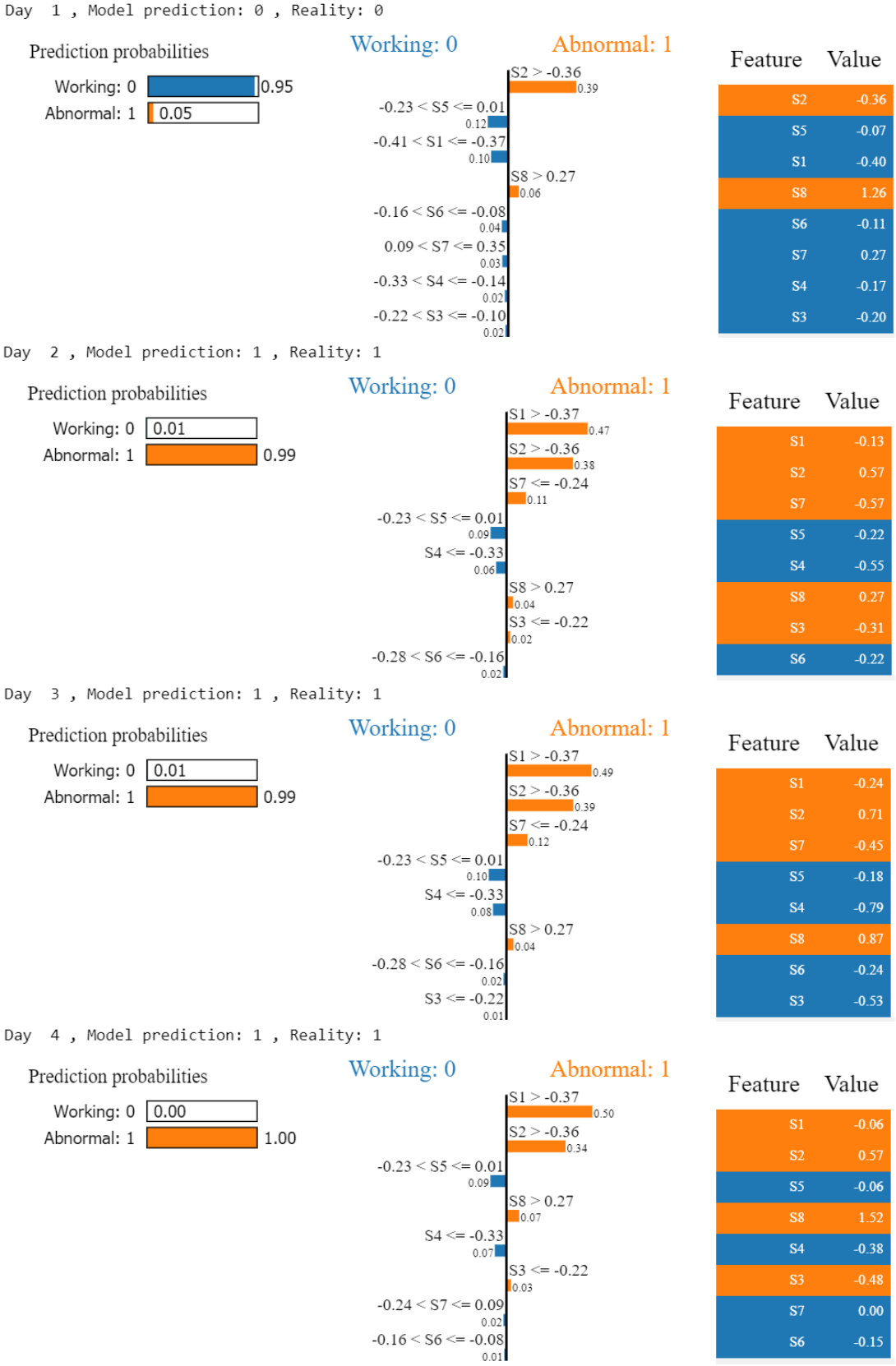



Figure 8: The output of LIME. Between day 2 & day 4 a valve breakage happened. The abnormality is affecting valve pair S2 (detected visually and in the report message), which matches the second explanation given by LIME.

The valve breakage in this period happened between day 2 & day 4. The abnormality affected valve pair S2, which was partly written in the report message and detectable visually from the sensor plots. The affected valve pair only matches the second explanation given by LIME.

Another example can be seen in figure 14 in the appendix. On day 2 and day 3 a valve breakage happened. This abnormality affected pairs S1 and S2, what we detected visually when plotting the sensor value. Only S1 is captured by LIME, but not S2, instead other sensors are said to have an abnormality. Interesting is also day 1 which is a "Working" day, but the top two explanations of LIME are pointing towards "Abnormality".

We checked all explanations of valve breakage events in the training data to see whether the sensor pair(s) explaining the abnormality according to LIME matched our information from the report or, if that information is missing, our visual analysis. Unfortunately this is not always true and the explanation during one abnormal period changed sometimes, see for example figure 14 in the appendix.

As an extension of LIME, Submodular Pick (SP-LIME) [12] tries to give a global understanding of the model by explaining a diverse, representative set of instances. We also looked at those results for different numbers of instances. In figure 9 the result for two instances is shown. The explanations for the instances classified as "Working" contained mostly features indicating an abnormal instance. It seems as if the linear model does not learn the underlying structure of our black box model very well.

One possible reason why LIME does not perform well is that it assumes linear behavior of the machine learning model locally, and we might have very nonlinear data. Another reason might be a bad choice of the kernel width of the exponential smoothing kernel. The best kernel width choice can only be detected by lengthy experiments. Another downside of LIME is that the explanations take up a lot of storage space, so it is very costly to store the explanation of every day of one year. Fortunately, the next method we tried, SHAP, worked better and takes up a lot less storage space.

## 4.2 SHapley Additive exPlanations (SHAP)

SHAP is based on the Shapley Value [18], a game-theoretic method for a fair allocation of the output among all members of a coalition. It is computed as the average marginal contribution of a member across all possible coalitions. Let N be a set of n players and v be a characteristic function, i.e. let v(S) be the worth of coalition S. Given a coalitional game $(v, N)$ the Shapley value is given by

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(n - |S| - 1)!}{n!} \left(v(S \cup \{i\}) - v(S)\right).$$

The idea for the usage of the Shapley Value in explaining a black box model is to interpret each feature value as a player in a coalition game and the prediction as payout. Then the Shapley Value of a feature is its average contribution to the prediction in different

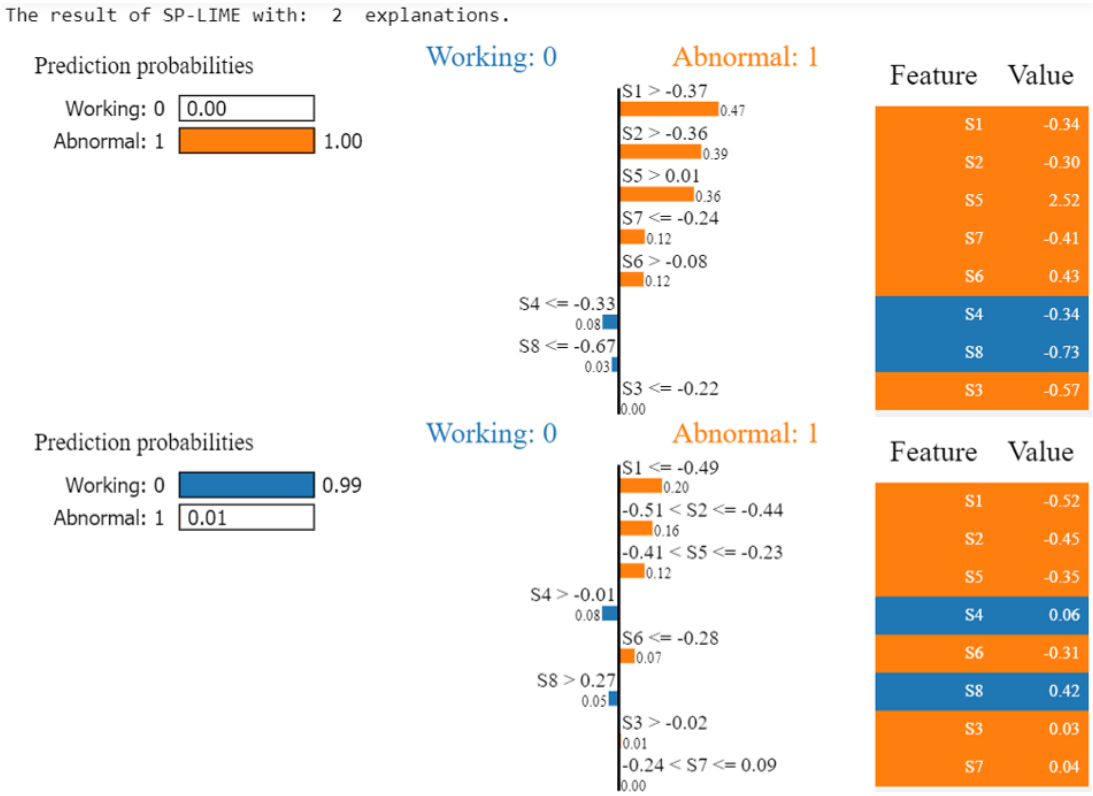



Figure 9: The output of SP-LIME when two representative instances shall be picked.

coalitions [10] and the prediction is explained as a game played by the feature values.

The biggest advantage of the Shapley value is that it is the only explanation method with a solid theory (see [18]), giving the explanation a reasonable foundation. It requires a lot of computing time as there are $2^k$ possible coalitions of the feature values, therefore in general approximations are needed. We used the approximation method KernelSHAP.

**Theory** KernelSHAP was presented in [8] and is an alternative, kernel-based estimation approach for Shapley values inspired by local surrogate models. It unifies six other XAI methods, one of them being LIME. This following explanation is taken largely from [10].

With the explanation model $g$ and $z' \in \{0,1\}^M$, SHAP gives the explanation by

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j.$$

Here, $M$ is the maximum coalition size and $\phi_j \in \mathbb{R}$ is the Shapley Value of feature $j$. The $z'$ describe the coalition of the instance: when the corresponding feature value is "present", then $z'$ has an entry of 1, if it is "absent", then $z'$ has an entry of 0.

Now for the instance $x$ that shall be explained it holds

$$f(x) = g(x') = \phi_0 + \sum_{j=1}^{M} \phi_j.$$

KernelSHAP estimates the Shapley Value $\phi_j$ of each feature value for an instance $x$ in the following way:
First, the coalitions $z'_k \in \{0,1\}^M$ are sampled with $k \in \{1,\dots,K\}$, where 1 means that the feature is and 0 means that the feature is not in the coalition. We then feed the input corresponding to the coalition $z'_k$ to our black box model and get its prediction. The weight for each $z'_k$ is computed via the SHAP kernel

$$\pi_x(z') = \frac{M-1}{\binom{M}{|z'|}\,|z'|\,(M-|z'|)}.$$

A weighted linear model is then fitted returning the Shapley values $\phi_k$. The SHAP kernel computes the weight the coalition would get in the Shapley value estimation. It has been shown in [8] that linear regression with this kernel weight indeed yields Shapley values.

The main difference of KernelSHAP to LIME are the weights of the instances in the linear model. Thus differently to LIME, the Shapley value does not give the change in prediction when we would remove the feature from the model, but the average contribution of a feature to the prediction in different coalitions.

**Application in the Model** We applied the implementation of KernelSHAP (implemented in the shap package [7]) to model (1) described in 3.7, where no outside temperature removal was done and the prediction was restricted to cylinder 1 and 2.

The result of SHAP can be seen in figure 10. Here, the classes "Working" and "Abnormal" are indicated by 0 respectively 1 and the model output is the probability for "Abnormal". The explanation of one day shows how much each feature is contributing to push the model output from the base value (the average model output over the training dataset we passed) to the model output. Features pushing the prediction towards "Abnormal" are shown in red, those pushing the prediction towards "Working" are in blue.
The valve breakage happening between day 2 & day 4 affected the valve pair S2, which we detected visually from the sensor plots. We can see in figure 10 that this matches the top explanation given by SHAP.

SHAP is also able to explain several instances is one plot, see figure 11. This is done by computing the Shapley Values for all instances, rotate the single explanations by 90 degrees, and then stack them horizontally.
In figure 11 another example of SHAP are shown. In this period sensor pair S5 was affected. Again, this was detectable visually from the sensor plots. This matches the top explanation given by SHAP.
In figure 15 in the appendix the output of SHAP is shown. A valve breakage happened from day 3 until day 14 and the sensor pairs S1 and S2 were affected. Again, this was

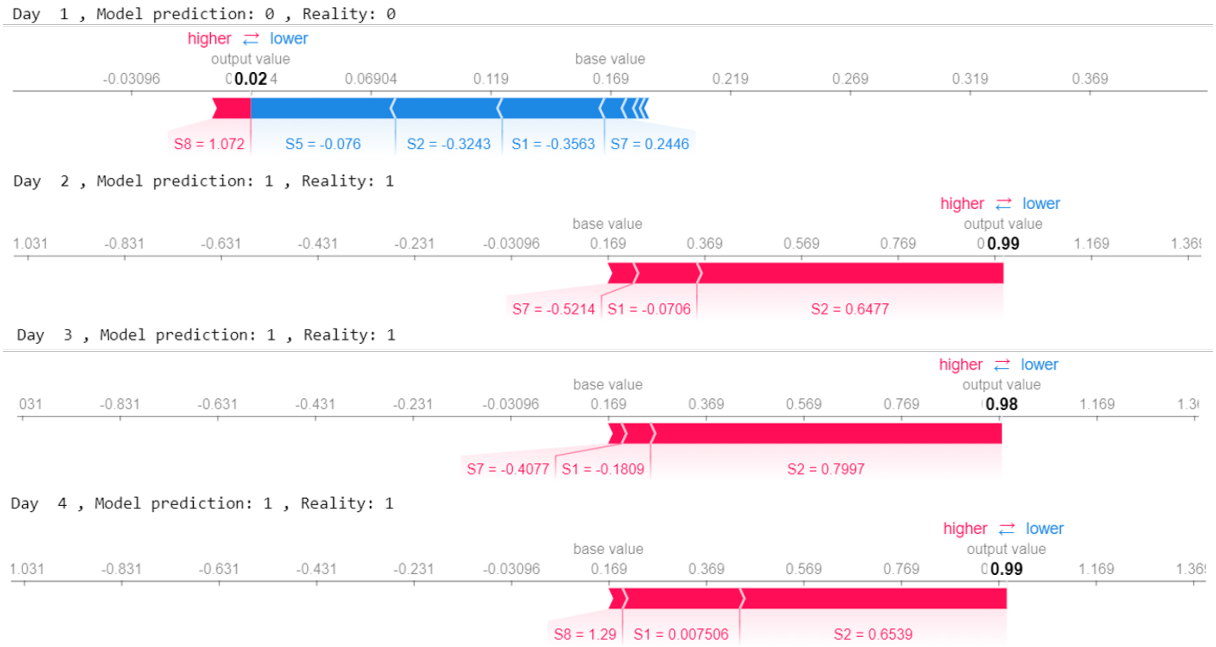



Figure 10: The output of SHAP. From day 2 to day 4 a valve breakage happened, affecting sensor pair S2.

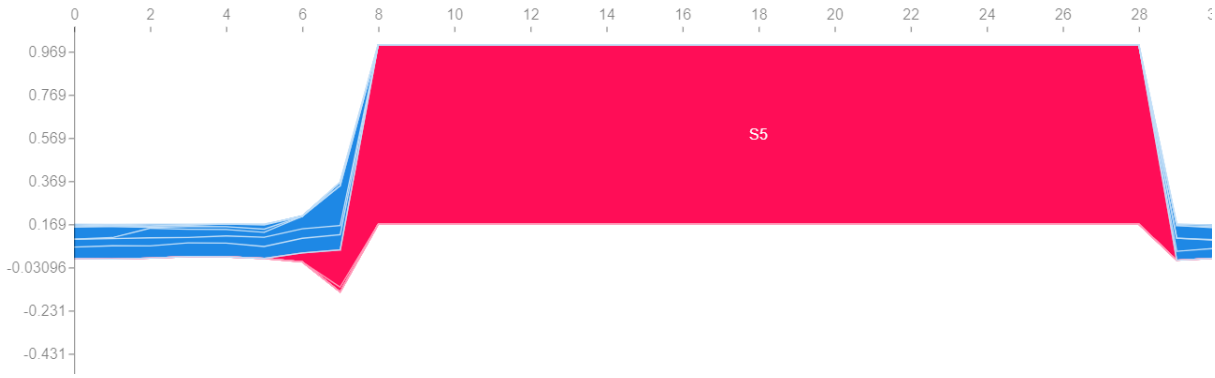



Figure 11: The output of SHAP showing S5 was affected.

detectable visually from the sensor plots. Interestingly, first S1 is the sole explanation for the valve breakage and S2 arises after some days. Since the valves affect each other and the report messages do not indicate which valve pair broke on which day, this is a conceivable scenario.

We looked at all explanations of valve breakages in the training data. SHAP always indicated the broken valve pair(s) in the explanations that we found visually and (if available) matched the report message. We are therefore confident our model does not only have a very high accuracy but can also indicate the broken valve pairs with SHAP.

# 5 Conclusion

In the first part of this project, we successfully chose a dataset appropriate for building a black box model and XAI and were able to build a linear model as benchmark and on RNN as well as extend one CNN. We learned a lot about different approaches to predictive maintenance and gained first experience with tensorflow, a new tool for all of us.

At the end of November, a new dataset from one of the clients of Horváth & Partners GmbH was made available to us. We were able to experience all exciting moments and challenges of a real-world data analysis task: we had to identify relevant sensors, gain technical knowledge, wait for information from the client and face all phases of building a model. During all exploratory data analysis, data preprocessing, data labeling, removal of outside temperature influence and downtimes as well as deciding on the actual architecture of the model, we learned that no data analysis task is the same and many innovative ideas and persistence are needed to arrive at the final goal: a highly accurate model.

Out of a large variety of XAI methods we applied two to our final model of the client's dataset. The topic of explainable AI is a quickly growing subject since AI methods in business applications need to fulfill laws and regulations. After reading a lot about the different methods, the application of two of them to our model made them tangible. We are especially happy that our model is not only highly accurate but we were able to verify it with SHAP.

We are proud that the client is pleased with our proof of concept models, and he is interested to use them for other similar machines in his site to predict valve breakages and minimize downtimes.

# Bibliography


[1] BackBlaze. *Hard Drive Test Data (kaggle challenge)*. Available at `https://www.kaggle.com/backblaze/hard-drive-test-data/data`. [Accessed October 2019]. Nov. 2016.

[2] Matous Cejnek. *Padasip Documentation, github*. Available at `http://matousc89.github.io/padasip/index.html`. [Accessed December 2019]. Aug. 2017.

[3] Finale Doshi-Velez and Been Kim. "Towards a rigorous science of interpretable machine learning". In: *arXiv preprint arXiv:1702.08608* (2017).

[4] Hannes Knobloch, Adem Frenk, and Wendy Chang. *GitHub repository long-live-the-battery*. Available at `https://github.com/dsr-18/long-live-the-battery`. [Accessed November 2019 with the last commit on 2 Jul]. July 2019.

[5] Hannes Knobloch, Adem Frenk, and Wendy Chang. *Predicting Battery Lifetime with CNNs*. Available at `https://towardsdatascience.com/predicting-battery-lifetime-with-cnns-c5e1faeecc8f`. [Accessed November 2019]. Sept. 2019.

[6] Zhou Lu et al. "The expressive power of neural networks: A view from the width". In: *Advances in neural information processing systems*. 2017, pp. 6231–6239.

[7] Scott M Lundberg. *Documentation of shap package*. Available at `https://shap.readthedocs.io/en/latest/`. [Accessed January 2020]. 2018.

[8] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in neural information processing systems*. 2017, pp. 4765–4774.

[9] *Maintenance KPIs and maintenance metrics*. Available at `https://www.fiixsoftware.com/important-metrics-maintenance-department/`. [Accessed December 2019].

[10] Christoph Molnar et al. "Interpretable machine learning: A guide for making black box models explainable". In: *E-book at https://christophm.github.io/interpretable-ml-book, version dated* 10 (2018).

[11] Marco Tulio Ribeiro. *Documentation of lime package*. Available at `https://lime-ml.readthedocs.io/en/latest/`. [Accessed January 2020]. 2016.

[12] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why should i trust you?" Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.

[13] Matteo Rulli and Luca Bixio. *Accelerating IoT Big-Data Analytics and Enabling insights Engineering in Industry 4.0*. Available at `http://www.datascienceseed.com/wp-content/uploads/2019/02/20190207-FlairBit-Challange.pdf`. [Accessed October 2019]. Feb. 2019.

[14] Sule Selcuk. "Predictive maintenance, its implementation and latest trends". In: *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 231.9 (2017), pp. 1670–1679.

[15] Kristen A Severson et al. *124 Lithium-ion data set*. Available at `https://data.matr.io/1/projects/5c48dd2bc625d700019f3204`. [Accessed October 2019]. Mar. 2019.

[16] Kristen A Severson et al. "Data-driven prediction of battery cycle life before capacity degradation". In: *Nature Energy* 4.5 (2019), p. 383.

[17] Kristen A Severson et al. *Supplymentary Infomation data-driven prediction of battery cycle life before capacity degradation*. Available at `https://data.matr.io/1/projects/5c48dd2bc625d700019f3204`. [Accessed October 2019]. Mar. 2019.

[18] Lloyd S Shapley. "A value for n-person games". In: *Contributions to the Theory of Games* 2.28 (1953), pp. 307–317.

[19] Taarifa and Tanzanian Ministry of Water. *Pump it Up: Data Mining the Water Table*. Available at `https://www.drivendata.org/competitions/7/pump-it-up-data-mining-the-water-table/page/24/`. [Accessed October 2019]. Sept. 2019.

[20] European Union. *Ethics guidelines for trustworthy AI*. Available at `https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai`. [Accessed November 2019]. Apr. 2019.

[21] Bernard Widrow. *Adaptive Signal Processing*. Pearson, 1985, pp. 307–317. ISBN: 0130040290.

# Appendix

| | Additional CNN features |
|---|---|
| 1 | Minimum $\Delta Q_{(i+20)-i}(V)$ |
| 2 | Variance $\Delta Q_{(i+20)-i}(V)$ |
| 3 | Skewness $\Delta Q_{(i+20)-i}(V)$ |
| 4 | Kurtosis $\Delta Q_{(i+20)-i}(V)$ |
| 5 | Slope of linear fit to discharge capacity |
| 6 | Intercept of linear fit to discharge capacity |
| 7 | Average discharge time |
| 8 | Minimum internal resistance |
| 9 | Difference internal resistance of 20 cycles |

Table 4: Addition features for battery dataset CNN

| **RNN Model** | | | |
|---|---|---|---|
| **Layer (type)** | **Output Shape** | **Parameters** | **Connected to** |
| IR (InputLayer) | [(None, 20, 1)] | 0 | - |
| QD (InputLayer) | [(None, 20, 1)] | 0 | - |
| Discharge-time (InputLayer) | [(None, 20, 1)] | 0 | - |
| concat-scalar (Concatenate) | (None, 20, 3) | 0 | IR[0][0], Discharge-time[0][0], QD[0][0] |
| recurrent (LSTM) | (None, 128) | 67584 | concatscalar[0][0] |
| dropout-lstm (Dropout) | (None, 128) | 0 | recurrent[0][0] |
| hidden (Dense) | (None, 32) | 4128 | dropout-lstm[0][0] |
| output (Dense) | (None, 2) | 66 | hidden[0][0] |

Table 5: RNN architecture with 71,778 trainable parameters in total

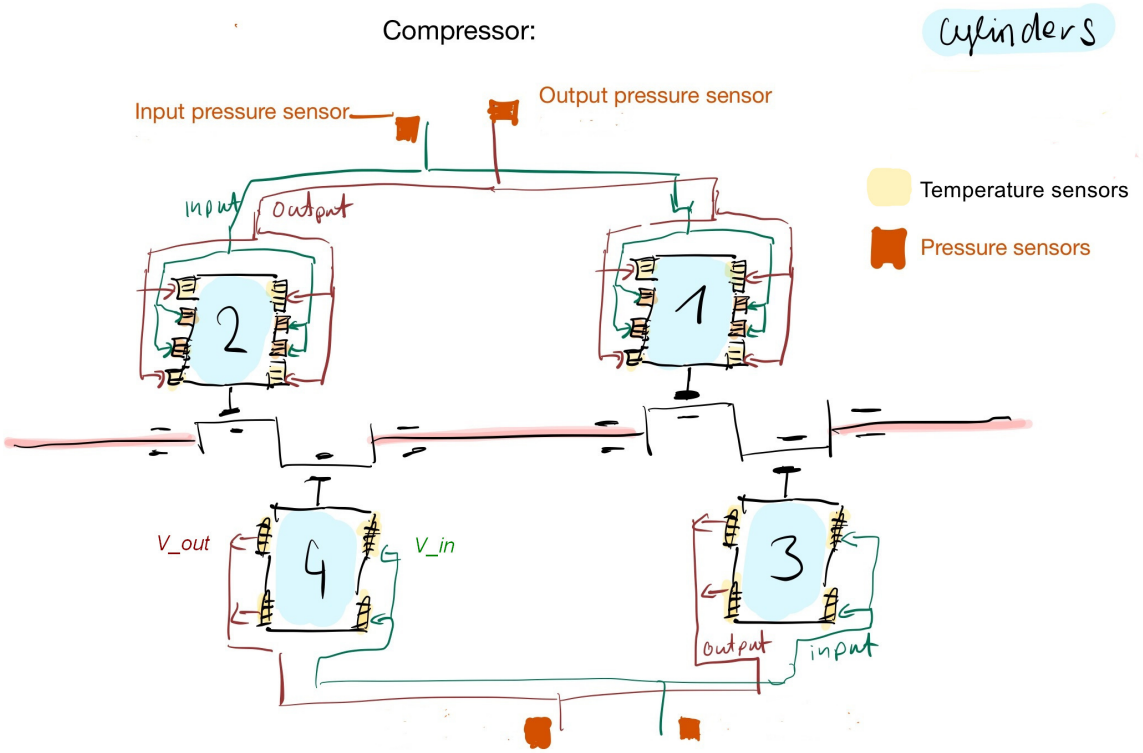



Figure 12: Diagram of the compressor

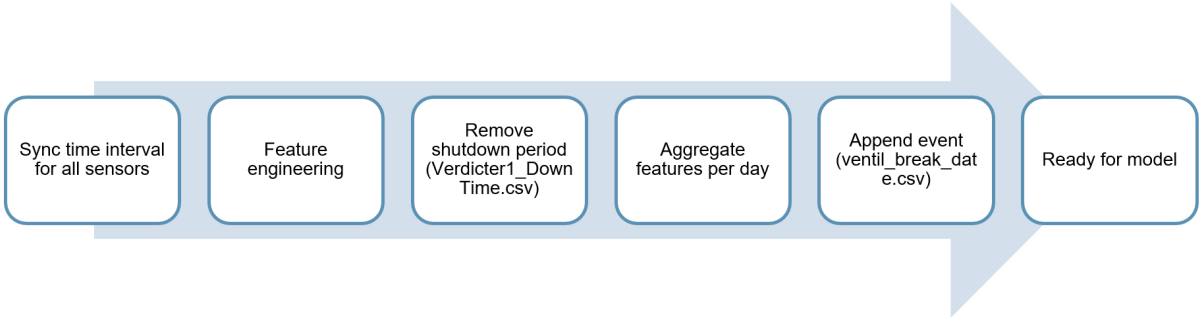



Figure 13: Important preprocessing steps of models' features

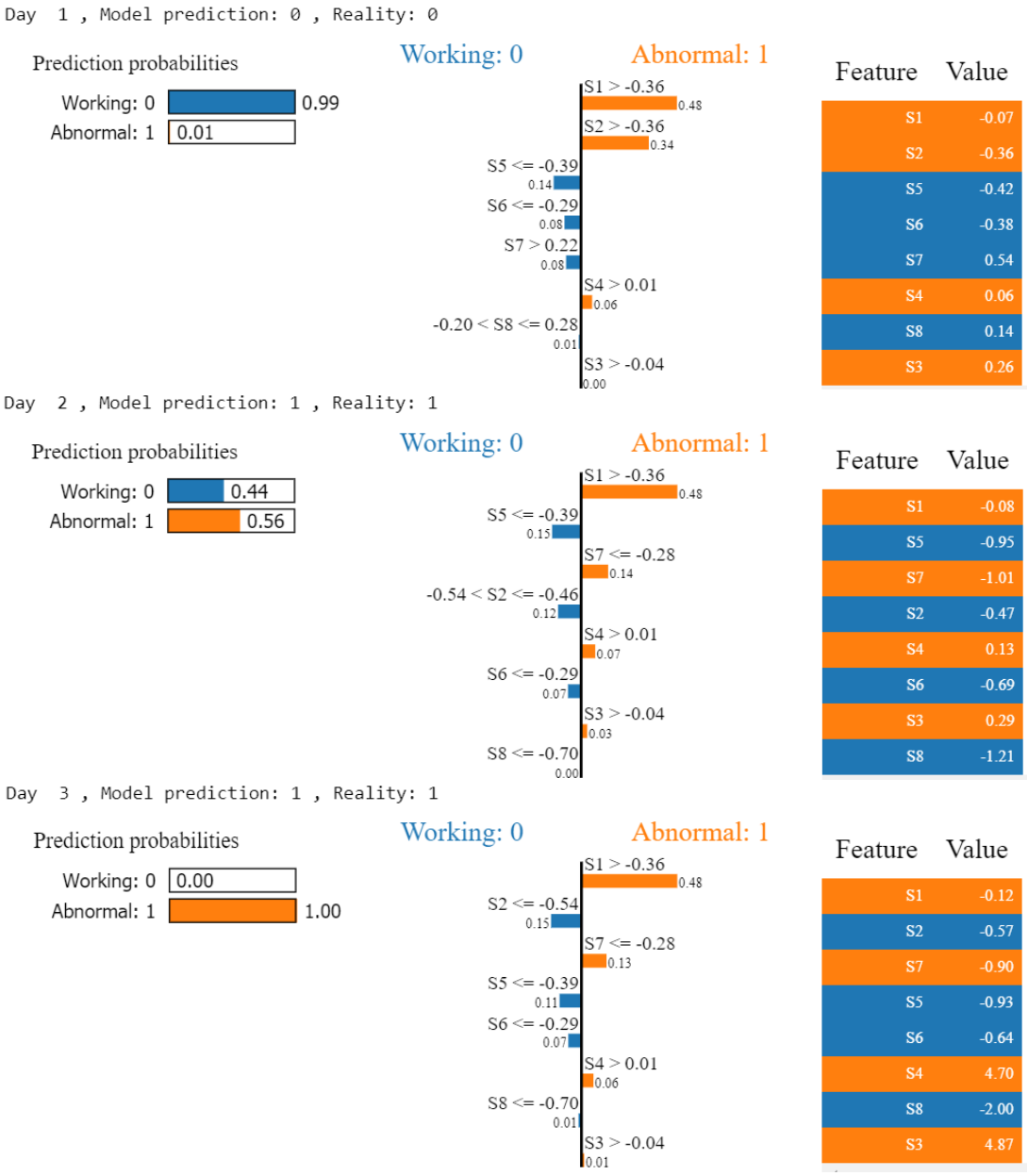



Figure 14: The output of LIME for three consecutive days. On the 2nd and 3rd day a valve breakage happened. The abnormality is affecting valve pairs S1 and S2 (detected visually and in the report message), which only partly matches the top explanations given by LIME.

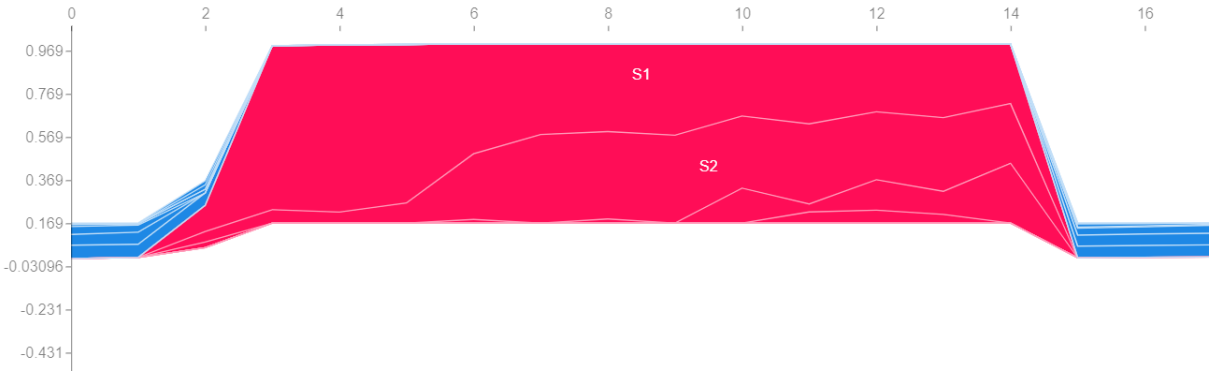



Figure 15: The output of SHAP. From day 3 until day 14 a valve breakage happened, affecting sensor pairs S1 and S2. This matches the top explanations given by SHAP.