Abstract—We present a reformulation of the regression and classification, which aims to validate the result of a machine learning algorithm. Our reformulation simplifies the original problem and validates the result of the machine learning algorithm using the training data. Since the validation of machine learning algorithms must always be explainable, we perform our experiments with the kNN algorithm as well as with an algorithm based on conditional probabilities, which is proposed in this work. For the evaluation of our approach, three publicly available data sets were used and three classification and two regression problems were evaluated. The presented algorithm based on conditional probabilities is also online capable and requires only a fraction of memory compared to the kNN algorithm.

I. INTRODUCTION

Neural networks, decision trees or support vector machines are used in many industrial areas today. In sensor technology and data analysis these machine learning methods are used to convert a multitude of complex measured values and data into useful information [1]–[10]. This includes position and orientation [11]–[14], monitoring of biometric data [4], [15]–[19], control of robots [13], [14], [20], [21], malware analysis [22]–[24], credit rating [25]–[28], visualization [29]–[32], eye movement evaluation [33]–[39] and much more. The ever-increasing number of applications for machine learning methods is due to the fact that they are very economical to use compared to the classical algorithms which usually demand for a more detailed knowledge about the modeled system and are less generic. Another advantage of machine learning methods is the steadily improving performance compared to classical algorithms. As a result, new areas where automation through machine learning methods is gaining ground are constantly growing.

A disadvantage of machine learning methods is that they cannot be easily explained and validated [40], [41]. In many areas, where living beings are involved and safety standards must be guaranteed, the application of complex procedures is not possible because their reliable function cannot be proven. Of course, there are exceptions such as the k Nearest Neighbour method (kNN) or probabilistic models such as the Gaussian Mixture Models (GMM). Unfortunately, these methods lack flexibility to work well for all problems and for example in case of the kNN, huge training datasets result in a computationally expensive classifier.

The range of application of an automatic online validation for machine learning algorithms [5] is manifold since classical algorithms are already replaced by machine learning algorithms [42]–[54]. An important point here is of course the validation of the output of the machine learning model. The most common areas of application are online measurement systems [55], [59] where data is collected via sensors from the areas of environmental [57], [58], biological [59], [60] or technical systems [61], [62]. Thus, these measuring systems are used for monitoring, control and long-term data acquisition, although in most cases they are not safety-critical applications. Nevertheless, a validation of the result is desirable, because this way a new run can be carried out or the user can be informed that the result is probably wrong and can thus exclude it for a data analysis. Also, the online validation can indicate defective sensors, because the output is certainly no longer valid for defective sensors. In the case of safety-critical applications it is even mandatory to validate the model and in case of very high safety requirements like a nuclear plant it is even mandatory to prove the functionality of the model.

In this work, we address the validation and explainability of the results of complex machine learning algorithms. For this purpose, we present an additional reformulation of the general problem definition for machine learning methods, which is described in detail in Section III.

Our contributions to the state of the art are as follows:

1. Reformulation of the general problem definition for machine learning methods.
2. Provable and explainable validation for machine learning algorithms in general.
3. Realization of our proposed validation approach using conditional probabilities and a kNN model with feature-based qualitative metrics.
4. Empirical evaluation of our approach to classification and regression problems on public data sets.
II. RELATED WORK

The first approach for Neural Network validation was published in \cite{63}, \cite{64}. As with our approach, the data is considered as a distribution and a manually defined threshold is used to decide whether the data is valid or not. The biggest difference to our approach is that we do not have to set thresholds, we formulate an additional quality metric per data input value, and we formulate the classes and quantified regression steps as conditional probability. In \cite{65} the authors used the difference in the responses of different neural network architectures trained on different subsets of the training data. If the difference between the model responses was too large, an invalid response can be assumed. A disadvantage of this approach in practical application is that it does not trace back to a cause of the error regarding the input data. A list of restrictions for neural networks and machine learning algorithms in general is given in \cite{66}, \cite{67}. One of the main problems that neural networks face is that they often lack a kind of white-box view of behaviour \cite{66}, \cite{67}. A layered validation and verification of online adaptive neural networks was presented in \cite{68}. Here, the input and output of the neural network are monitored and confidence intervals are calculated using input data. If the input data differs too much from the previous seen input data, the output is considered uncertain. Since this work is in progress, as stated by the authors, the exact calculation was not explained in detail.

The latest approaches to the validation of neural networks attempt to validate every single neuron in a network. This is done by reformulating it into a linear programming formalism which is then checked using box constraints \cite{69}, \cite{70}. Theoretically, it is the best approach but since it cannot be used online, only given input and output conditions are tested, and not all neurons can be validated in general, a lot of research is needed before these approaches can be applied in practice. In addition, \cite{69}, \cite{70} are limited to neural networks.

III. METHOD

In this work, we deal with the validation and explainability of the results of complex machine learning algorithms. For this purpose, we present an additional reformulation of the general problem definition for machine learning methods. In the case of classification, this problem is given by $M(F^n) \rightarrow \mathbb{N}$, where $M$ is the function which is calculated by the machine learning method, $F^n$ is the $n$-dimensional data or sensor values and $\mathbb{N}$ is the class output. To validate the output class $k \in \mathbb{N}$, we suggest a "backprojection", which is a binary classification. Thus the "backprojection" can be represented as $V(F^n, M(F^n)) \rightarrow \{-1, 1\}$. This classification indicates whether the output class $M(F^n)$ represents a valid solution for the given input data $F^n$. The advantage of this formulation is that the complex multiclass problem can be mapped to a simpler binary classification problem. Therefore, simple machine learning methods for $V$ can be used without having to expect large losses in the accuracy of the prediction method $M$.

In the case of regression, the underlying problem is $M(F^n) \rightarrow \mathbb{R}^m$. To make our procedure applicable to such a problem, we use quantization on the real output $\mathbb{R}^m$ to form a multiclass classification problem. The formula is $Q(\mathbb{R}^m) \rightarrow \mathbb{N}^m$ where $Q$ is the quantization and each $m$ can have a different division. With this quantization, the "backprojection" can be defined as $V(F^n, Q(M(F^n))) \rightarrow \{-1, 1\}^m$. This means that each output value $m$ of the machine learning model is considered either as valid 1 or invalid −1.

Since in real or industrial applications the pure class assignment is usually not sufficient, a quality signal can be determined by the distance to the individual classes (\{-1, 1\}) and this can also be mapped to the value range [0, 1]. In the case of a kNN for the backprojection, the distance of the k selected neighbors to the input data would be exactly this distance. Mapped to the value range [0, 1], this would allow the statement how similar the input data is to already known data. Thus, the result of the machine learning process can be easily explained using the training data. Since a kNN requires a lot of memory for the individual training samples and is expensive to calculate with increasing amounts of training data, we describe the backprojection by means of a conditional probability in the following. In addition, the individual input data streams are mapped to distributions and thus allow to evaluate the quality of individual data streams in general.

Figure 1 shows the process of our novel approach for the online validation of the results from machine learning approaches. In Figure 1 we selected a neural network as an instance. The upper part (orange) is the simple execution. Here the network is applied to sensor data and delivers a result.

The lower part (green) is our approach to validate the result. In case of a regression, the results of the neural network are quantified to form conditional probability distributions. This is already given for the classification. Each quantification level or class forms an index to probability distributions calculated on the sensor data. Since these form a high-dimensional vector in most cases, either a dimension reduction must be applied or each individual sensor value must be represented as a probability distribution. Using the quantified result of the network, it is now possible to calculate from the distributions how certain one can be that the input data matches the data already seen (training data) with respect to the result. This gives you the information whether you have already evaluated the net on similar data and the result is the same or not. It makes it possible to validate online whether the sensor values match the data that was used for the approval of the algorithm. Another advantage of the distributions on the sensor data is the quality of the distribution. In high-dimensional input vectors, there may be single values that do not contribute to certain results. Therefore the distribution has no significance or a high degree of uncertainty for the result. This can be measured by the difference of the integrals of the raw data and the distribution. This measure of uncertainty in combination with the assignment to quantified results allows to weight the contribution of individual distributions to the validity differently.
In the following, the whole procedure is described mathematically. During the operational mode, we aim to compute a conditional probability

\[ P_{\text{NET}}^{i,j}(x_i,t|y_j,t) \] (1)

which is the probability of \( x \) given the output \( y \) at time \( t \) of the machine learning approach. Since the input raw values \( x \) and the output \( y \) are vectors, we consider each index \( i \) of the input vector \( x \) and each position \( j \) of the output vector \( y \) separately. \( y \) contains the discretized sensor values where the index \( j \) labels the intervals. The validity of the output is judged by comparison of the distribution in Equation 1 with a second distribution which was fitted beforehand against the training dataset:

\[ P_{\text{GT}}^{i,j}(x_i,t|y_j,t) \] (2)

Equation 2 represents the calculated distribution for \( y_j \), which was fitted using the input data \( x \), and the ground truth labels together with the network output. Therefore, this conditional probability distribution can be understood as a reliability function of the network. This can be done either as histogram or using a fitting algorithm to an formally defined function like a gauss distribution. The formally defined function itself is freely selectable, but care must be taken that the difference between the output of the formally defined distribution and the raw distribution is as small as possible. We continue with the calculation of the normalized intersection of Equations 1 and 2:

\[ Q_{j,i} = \frac{P_{\text{NET}}^{i,j}(x_i,t|y_j,t) \cap P_{\text{GT}}^{i,j}(x_i,t|y_j,t)}{\sum_{i=1}^{N} \max(P_{\text{GT}}^{i,j}(x_i,t|y_j,t))} + \epsilon \] (3)

Equation 3 determines the quality of the distribution and corresponds to the Jaccard Index for area comparisons. This means that the more similar the two distributions are, when superimposed, the higher is the quality. For the comparison of distributions \( P,Q \) the Kullback-Leibler divergence \( D(P,Q) \) is normally used, and is characterized by an asymmetry \( D(P,Q) \neq D(Q,P) \). As we have no preferred ordering and store the distributions as histograms, it makes sense to use the symmetric Jaccard Index for our implementation. The normalization leads to a maximum value of 1 and a minimum value equal to 0. \( \epsilon \) is needed to prevent a division by 0. An advantage of this formulation is that it is very sensitive to gaps in the distribution and results in a poorly rated quality if the data base is small. With these three equations we can formulate the overall evaluation of new input data \( x \) with an output value \( y \) at time \( t \).

\[ V(j,t) = \frac{\sum_{i=1}^{N} Q_{j,i} \times P_{\text{GT}}^{i,j}(x_i,t|y_j,t)}{\sum_{i=1}^{N} \max(P_{\text{GT}}^{i,j}(x_i,t|y_j,t))} \] (4)
Equation 4 describes the computation of the validity of each $j$th response of the machine learning algorithm separately, which is illustrated as neural network in Figure 1. The computation consist of the quality of the features ($Q_{j,i}$) and the reliability of the network output per feature ($P_{j,i}^{GT}(x_{i,t}|y_{j,t})$). For normalization and to ensure numerical stability, the sum between the product of the quality of the features and the reliability of the network output per feature is divided by the maximum values of the reliability of the network output per feature ($P_{j,i}^{GT}(x_{i,t}|y_{j,t})$) since those values are the maximal outcomes of the product ($Q \in [0, 1]$).

For a large validity $V(j,t)$, the scalar product $\sum_{i=1}^{N} Q_{j,i} \cdot P_{j,i}^{GT}(x_{i,t}|y_{j,t})$ has to be large, which is the case for a precomputed distribution $P_{j,i}^{GT}(x_{i,t}|y_{j,t})$ that coincides well with the one based on the model data alone $P_{j,i}^{NET}(x_{i,t}|y_{j,t})$. Then, $Q_{j,i}$ and $P_{j,i}^{GT}(x_{i,t}|y_{j,t})$ are similar, which maximizes the numerator in Equation 4. The normalization is done using the sum of maxima of all the included distributions which leads to a preference of homogeneous distributions with small maxima. The validity $V(j,t)$ has a maximum value of 1 and a minimum value of 0. This quality signal can be used to measure whether the input data follows the data used to validate the algorithm or not. Thus, it is possible to make online statements whether the response of the neural network is reliable and has been tested and is therefore explainable and comprehensible. Alternative formulations for Equation 4 can use the median of all computed values or redefine a minimum of quality, by using a threshold.

The formulation of Equation 4 is limited by the need for sufficiently many data points which correspond to the assigned output and provide a high quality according to Equation 5. If those data is not given, no statement can be made with regard to the input data, since these cannot be represented as a distribution either.

To circumvent this limitation we follow the principle of divide and conquer. We determine a large number of distributions that fit as well as possible to a local value of the output. This way we simplify the complexity of the whole distribution to many smaller distributions. In addition, we also divide the nonlinear output into ranges (quantization) or use the predefined classes. Since the individual outputs of the machine learning algorithm are also considered separately, the validity can be evaluated separately for each output, but also in total for the whole network (Equation 5).

$$V_{all} = \frac{\sum_{j=1}^{M} V(y_j)}{M} \quad (5)$$

Equation 5 describes the evaluation of the whole network over all outputs $j$ where the total number of outputs is $M$. This is a simple average where each output can be weighted to obtain a weighted average validity ($V_{all, w} = \frac{\sum_{j=1}^{M} w_j \cdot V(y_j)}{\sum_{j=1}^{M} w_j}$) with $w_j$ as weight per output.

IV. EVALUATION

In this part of the work, different non-linear machine learning approaches (neural networks, gradient boosted decision trees, and bagged decision trees) are applied to three public data sets. For each machine learning approach we evaluated different model sizes, but only a small part is shown here. The remaining evaluations are in the supplementary material. For the evaluation, we used three classification challenges and two regression challenges. For the validation, we show the results of the both approaches discussed in the last section: The probabilistic approach, and the kNN approach based on the reformulation and histograms as distributions. First, we describe the public data sets used.

Beach Water Quality If This data set contains the sensor data from the water quality of the Chicago Park District along Chicago’s Lake Michigan lakefront. It is recorded on six different beaches which are our target classes for the classification experiment. For regression, we used the wave height sensor response. In total it has 39,469 entries but we omitted all records where at least one value was missing. This was done to make all machine learning based approaches like neural networks applicable to the data. Therefore, we used 10,034 records for our evaluation.

Classify Gestures by Reading Muscle Activity This data set was recorded using a MYO armband which records the signals from eight EMG (Electromyography) sensors. Eight of such recordings are connected (64 features per class) and linked to a muscle activity (Gesture classes were: rock - 0, scissors - 1, paper - 2, ok - 3). These gestures are the targets for our classification experiment. We did not use this data set for regression since the sensor values would not make much sense as targets. In total the data set has 11,678 records which we all used in our experiment.

Red Wine Quality This data set was published in [71]. It is related to red variants of the Portuguese “Vinho Verde” wine and contains eleven features like fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, etc., as well as a numerical quality measure. For classification, we used the numeric quality measure as target. For our regression experiment, we selected citric acid as target value. It is a highly unbalanced data set with 1.599 records. We used all of the records for our evaluation.

A. Evaluated Machine Learning Approaches

To test our approach extensively, we evaluated three machine learning algorithms in four different configurations each. The selected machine learning algorithms are neural networks, bagged ensemble of decision trees, and gradient boosted ensemble of decision trees. We selected those because all of these approaches are known to perform very well on nearly all data sets. For the neural network, we used 20, 30, 40, and 50 neurons as one hidden layer before the final output neuron. For the regression we used Levenberg Marquart backpropagation [73] for training. For the classification we used Scaled Conjugated gradients [73]. The bagged and boosted ensembles

1https://data.cityofchicago.org/Parks-Recreation/Beach-Water-Quality-Automated-Sensors/qmqz-2xku
Fig. 2. The first (top) visualization is a confusion matrix for the Gestures classifier accuracy. The bottom right corner of the confusion matrix shows the overall accuracy of the classifier. The three central plots show the validity signal per sample for correctly classified examples (left kNN, center Probability, and right re-implementation of [63]). For the incorrectly classified examples the validity signal per sample is shown in the three plots at the bottom in the same order.

were used with 5, 10, 15, and 20 decision stumps. For the classification with bagging, we used the standard randomized approach form random forests [74] and quantile regression forests [75] for the regression. The classification with boosting was done using totally corrective boosting [76], and for the regression we used least-squares boosting [77].

The validity signal was estimated using a kNN [78]. As an alternative using our proposed approach without explicit distribution fitting for the features, we used the estimated distribution from the data as histogram. This means that our quality of the distribution is always one (Equation 3). For the kNN we set K as the number of features in the data set and used euclidean as distance metric. In addition, we compared the distance to all known entries instead of an approximation with a tree structure. In addition, we compare our results with the previous work [63] to show the advantage of our algorithm.

The train and test split was done using a 50% to 50% split. This means we used 50% for training of the classifier and for the validation of them. This data was also used to train the validation algorithms. The other 50% are for evaluation only.

Since showing the complete evaluation of all the models for three classifications and two regression problems would exceed the content of this work, only the results of the neural network with 50 neurons in the hidden layer were included. All other results are found in the supplementary material.

B. Results

Figure 2, 3 and 4 show the classification results as well as the validation signal. The classification results are displayed as a confusion matrix for each data set (top plot). The three central plots show the validation signal on the correctly recognized classes per algorithm (left kNN, center Probability, and right re-implementation of [63]). Here it is desirable that the validation signal is as high as possible. As can be seen in all plots, the signals are high for both the kNN and for our conditional probability approach, but not for the [63] in Figure 2 and 4. Here the validation signal of the re-implementation has a significant impact on the classification accuracy.

Fig. 3. The first (top) visualization is a confusion matrix for the Beach Water Quality classifier accuracy. The bottom right corner of the confusion matrix shows the overall accuracy of the classifier. The three central plots show the validity signal per sample for correctly classified examples (left kNN, center Probability, and right re-implementation of [63]). For the incorrectly classified examples the validity signal per sample is shown in the three plots at the bottom in the same order.

Figure 3 shows that for some correctly recognized classes the validation signal of the kNN and Probability approach slips below a validity of 60%. This reduces the accuracy of the classifier. In contrast to the correctly recognized classes are the incorrectly recognized classes, which are shown in the three
In Figure 4, the first (top) visualization is a confusion matrix for the Red Wine Quality classifier accuracy. The bottom right corner of the confusion matrix shows the overall accuracy of the classifier. The three central plots show the validity signal per sample for correctly classified examples (left kNN, center Probability, and right re-implementation of [63]). For the incorrectly classified examples the validity signal per sample is shown in the three plots at the bottom in the same order.

In Figure 5 this looks a little different. Here we are trying to classify the beach based on the water sensor values. If we look at the validation signal in the bottom three plots, we see that for the kNN three entries and for the conditional probability approach two entries are wrongly considered valid. This is due to the similarity of the data, as it is possible that on some days the water quality on the beaches does not differ. The reason kNN scores worse is that the data are very unbalanced, as can be seen from the confusion matrix in Figure 3. In the case of correctly classified data, only one entry is considered invalid by our conditional probability approach and three entries by the kNN. This has the same reasons as for the incorrectly classified data. [63] provides worse results in both cases.

For the Wine Quality Classification in Figure 4, the confusion matrix again shows very unbalanced data. The validation, however, works very well for the wrongly classified data, where all three approaches do not consider a wrongly classified entry as valid. In the case of correctly classified data, four entries are considered invalid by the conditional probability approach and three entries are marked as invalid by the kNN approach.

<table>
<thead>
<tr>
<th>Output Class</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Class</td>
<td>0</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.4%</td>
<td>1.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>100%</td>
<td>100%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

Fig. 4. The first (top) visualization is a confusion matrix for the Red Wine Quality classifier accuracy. The bottom right corner of the confusion matrix shows the overall accuracy of the classifier. The three central plots show the validity signal per sample for correctly classified examples (left kNN, center Probability, and right re-implementation of [63]). For the incorrect classified examples the validity signal per sample is shown in the three plots at the bottom in the same order.


Fig. 5. The first (top left) visualization shows the absolute error (y axis) related to the target waive height value (x axis). The three following plots (kNN, Probability, and re-implementation of [63] in the same order) show the validity signal (y axis) to the error of the prediction (x axis). This means that the validity (y axis) should decrease, the higher the value on the x-axis will be.

For the evaluation of the regression, we have decided to quantify to 10 bins with an enlargement of the target value by a factor of 10. The maximum target value after enlargement is assumed to be 10. This applies to the regression of the wave height as well as to the citric acid. We used the value of 10 for quantification since the wave height are fixed values between 1 to 10 and the citric acid value range was from 4.6

Fig. 6. The first (top left) visualization shows the absolute error (y axis) related to the target waive height value (x axis). The three following plots (kNN, Probability, and re-implementation of [63] in the same order) show the validity signal (y axis) to the error of the prediction (x axis). This means that the validity (y axis) should decrease, the higher the value on the x-axis will be.
to 15.9. The enlargement factor of 10 was selected since the numerical accuracy for the citric acid value is 10⁻¹. The results are shown in Figures 5 and 6. The first plot shows the error distribution (y-axis) regarding the true magnitude on the x-axis (Figure 5 the wave height and Figure 6 the citric acid content). The three subsequent plots show the validation signal of the kNN, the conditional probabilities, and the re-implementation of [63]. As can be seen, the validation signal decreases with increasing error, which shows the desired behaviour of our approach. To express this numerically we accept an error of 0.1 (corresponding to 100 in Figure 5 and Figure 6 of the second, third, and fourth plots) since we have quantized the values to 10 and take 10 as the maximum value. If we take this error as a threshold value, we get that for citric acid kNN recognized 6.7% of all correct results as invalid and interpreted 1.1% of all wrong values as valid. For the conditional probability approach, 14.2% of the correct values are marked as invalid and 0% of the incorrect values as valid. In comparison to this the re-implementation of [63] marked 42.7% of the correct values as invalid and 13.2% of the incorrect values as valid. For the wave height with the same threshold value of 0.1, kNN marks 4.2% of the correctly recognized heights as invalid and 1.2% of the incorrectly recognized heights as valid. The conditional probability approach marks 4.3% of the correctly detected heights as invalid and 1.2% of the incorrectly detected heights as valid. The re-implementation of [63] marked 4.5% of the correct values as invalid and 4.2% of the incorrect values as valid.

V. CONCLUSION

In this work we have shown how to transform a multi class and regression problem into a two class problem with the classes valid or invalid. This two class problem is easier to solve and can be used to validate the result of a machine learning algorithm. For this purpose, an explanatory algorithm is necessary why we have evaluated a KNN for this purpose in this work. Since a KNN with a constantly growing amount of training data requires more and more memory and the computing time is increasing because more comparisons have to be made, we presented an alternative solution using probabilities. Both methods were evaluated on three public datasets where two regression problems and three classification problems were considered for a variety of machine learning algorithms and compared against a re-implementation of a state-of-the-art algorithm [63]. Our results show that our approach provides a high quality validity signal and works on a variety of problems. Both presented approaches can be used online and the probabilistic approach can be executed on a microcontroller even with large data sets. Together with the explanatory power of validation based on the training data, our approach represents a step towards machine learning algorithms in critical areas of application. Future work will go in the direction of sensor data acquisition regarding critical areas to further evaluate and validate our method for a real application.

REFERENCES


