IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 25, 2020, accepted June 27, 2020, date of publication July 2, 2020, date of current version July 15, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3006711

Building Uncertainty Models on Top of Black-Box Predictive APIs

AXEL BRANDO[1,2], DAMIÀ TORRES[2], JOSE A. RODRÍGUEZ-SERRANO[2], AND JORDI VITRIÀ[1]
[1]Departament de Matemàtiques i Informàtica, Universitat de Barcelona (UB), 08007 Barcelona, Spain
[2]BBVA Data and Analytics, 28050 Madrid, Spain

Corresponding author: Axel Brando (axelbrando@ub.edu)

This work was supported in part by the Industrial Doctorates Plan of Generalitat de Catalunya, in part by Banco Bilbao Vizcaya Argentaria (BBVA) Data and Analytics, and in part by the Universitat de Barcelona under Grant RTI2018-095232-B-C21 and Grant SGR 1742.

ABSTRACT With the commoditization of machine learning, more and more off-the-shelf models are available as part of code libraries or cloud services. Typically, data scientists and other users apply these models as "black boxes" within larger projects. In the case of regressing a scalar quantity, such APIs typically offer a `predict()` function, which outputs the estimated target variable (often referred to as $\hat{y}$ or, in code, `y_hat`). However, many real-world problems may require some sort of deviation interval or uncertainty score rather than a single point-wise estimate. In other words, a mechanism is needed with which to answer the question "How confident is the system about that prediction?" Motivated by the lack of this characteristic in most predictive APIs designed for regression purposes, we propose a method that adds an uncertainty score to every black-box prediction. Since the underlying model is not accessible, and therefore standard Bayesian approaches are not applicable, we adopt an empirical approach and fit an uncertainty model using a labelled dataset (x, y) and the outputs $\hat{y}$ of the black box. In order to be able to use any predictive system as a black box and adapt to its complex behaviours, we propose three variants of an uncertainty model based on deep networks. The first adds a heteroscedastic noise component to the black-box output, the second predicts the residuals of the black box, and the third performs quantile regression using deep networks. Experiments using real financial data that contain an in-production black-box system and two public datasets (energy forecasting and biology responses) illustrate and quantify how uncertainty scores can be added to black-box outputs.

INDEX TERMS Aleatoric uncertainty, deep learning, neural networks, regression problems.

I. INTRODUCTION

The success of machine learning in the real-world problems has led to the increasing commoditization of machine learning systems. Nowadays, more and more predictive models are available "off-the-shelf" as part of code libraries [35], [38], machine learning servers [10], cloud-based services [1] or inside domain-specific black-box software [39]. In other words, machine learning has become increasingly more accessible to users and developers who are not specialists in this field, but who wish to consume its predictive functionality. It has also become more commonplace to use a predictive system as a *black box* inside a larger engineering system [45].

In this paper, we consider regression models implemented as black boxes. The term *black box* denotes a predictive model that exhibits three properties: (1) it has been pre-trained, (2) it exposes a function (typically called `predict()`) to estimate the output value given a set of input values, and (3) its internal details, such as algorithm of choice, or dataset used for training, are not accessible or even known (typically the case with prediction APIs).

However, there are many situations of practical value where a `predict()` function is not enough, since we need to have access to a *range estimate* rather than a point estimate. This is the case when we want to output the confidence interval of the predicted value or assess the typical noise of the target variable. Generally speaking, data scientists encounter situations where they might be interested in the *distribution* of the target variable rather than just the "most probable" value.[1]

The associate editor coordinating the review of this manuscript and approving it for publication was Mohamad Forouzanfar.

[1]As an example, most regression classes in python's `sklearn` do offer a `predict()` function but not the means to get prediction intervals.

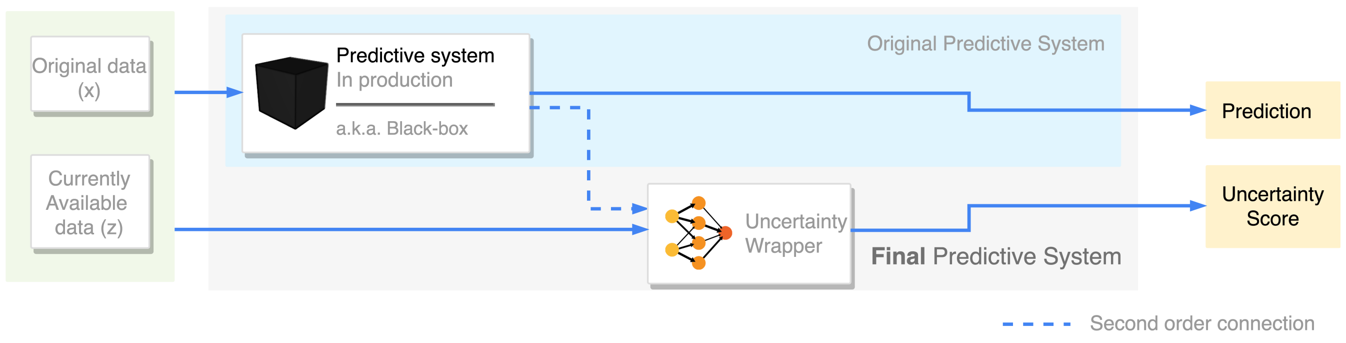


FIGURE 1. **Proposed method to upgrade any black-box predictive API (for regression) with an Uncertainty Score.**

While an alternative would be to move to white-box implementations of models that do handle the target variable distribution, such as Gaussian Processes [43] or Generalized Additive Models [5], *our aim here is to provide a solution when this is not possible*. Due to the applied nature of their work and delivery cycles, data scientists tend to be *consumers* of models rather than develop their own models from scratch. Our data science teams experience realistic situations in which substituting the black box is not viable or not desired, mainly because one of the crucial properties of the black box needs to be retained. Common examples of this are when the black box is a very reliable system (from a software engineering perspective), costly to replace or is required to fulfil a specific function in order to comply with regulations (e.g. medical [50] and some financial models, which sometimes need to comply with interpretability constraints). To sum up, we assume that the data scientist still needs the black box, but there is a need to "upgrade" it with uncertainty scores. In fact, this work was motivated by one of the data science teams needing to re-purpose an internal engine for forecasting – a case we discuss in the experimental section of this paper.

II. RELATED WORK

The black-box concept is normally used in contexts where the goal is to approximate a full model with another one that is considered as a white box [20], [36], i.e. aiming to decode or make accessible a complex model generating a new one that imitates this currently-in-use system. In this work, we have a different reason for considering the original model as a black box. Similarly to [33], [34] but applied to regression problems, the main goal here is not to generate a new model that imitates the previous one but to generate a new wrapper model that complements and maintains the original prediction with new information: the uncertainty score, as shown in Figure 1.

Not assuming the internal structure of the currently-in-use system implies several limitations. One of the most important is that approaches such as Bayesian optimization of the parameters or density estimation of the parameters are not applicable, as this would mean assuming that the original model is parametric, as is the case in [24]. However, our black-box system could be a non-parametric predictive system or even a handcrafted rule-based predictive system. In this context, Bayesian techniques such as MC-Dropout [17] or Bayesian neural networks [7], [21], [42], [48] or considering an ensemble of black boxes [29], [31] is not possible since we assume that we have a single fixed pointwise predictive system. In other words, the epistemic uncertainty that refers to not knowing the parameters distribution of the black-box model cannot be captured in the described scenario. Therefore, we focus on the other type of uncertainty - aleatoric - from the outset, which corresponds to the variability of possible correct answers given the same input, $p(y \mid x)$.

Aleatoric uncertainty is modelled using deep learning models for classification tasks [25], [32] and regression ones [8], [47]. However, not all aleatoric solutions can be applied to the uncertainty modelling of a black-box predictive system. The only applicable solutions are those that can disentangle the uncertainty prediction from the prediction of the response variable in a special manner described hereafter, where the response variable prediction can be substituted by the black-box prediction as shown in Figure 1.

III. PROBLEM STATEMENT

A. DEFINITION OF BLACK BOX

We define a *black-box predictor* (henceforth *black box*) as the implementation of a predictive model exposing a function of the form $\hat{y} = B(x)$. Here, x represents a set of inputs, and $\hat{y}$ represents a predicted quantity, which we will denote *prediction*. The fundamental assumption is that we do not know the functional form of B, and all we can do is call $B(x)$ and observe the result $\hat{y}$. Experienced data scientists can probably identify with this situation when using a model method implemented in an off-the-shelf library [35], [38], API, or cloud service – in many implementations, the function B is called `predict()`. To further frame our problem, let us discuss some considerations related to the inputs, output and assumptions regarding B in greater detail.

1) INPUTS TO B

The inputs x represent the attributes that would be passed to B. Let us consider the following two situations:

The case of *preserved input* appears when we have access to the actual values of x at the time we evaluate $B(x)$.

The case of *distorted input* occurs when we have access to the prediction of the black-box system $B(x)$ but not to its input x. For example, let us consider a public API to forecast electricity consumption in a given geographical area. While we can call the API, and get its *prediction*, in this case we do not have access to the input variables the model uses. However, we can still build some variables that describe the 'state' of the input so that this state is informative of the uncertainty. Indeed, later we will see some experiments where the available information (previous consumption values) is used as variables z to provide a reasonable input context and successfully forecast the uncertainty.

Finally, we should consider two different model configurations. First of all, when the input of the uncertainty wrapper is z, we refer to the model as a *First Order Decision model*, $\psi(z)$. Otherwise, when z and the black-box prediction $B(x)$ are both considered input attributes of our uncertainty estimator, we denote this model a *Second Order Decision model*, $\psi(z, B(x))$ (shown as a dotted arrow in Figure 1).

In experiments (Sec. VI), we consider *preserved* and *distorted* input problems and also compare *First* and *Second Order Decision models*.

2) OUTPUT OF *B*

The output, $\hat{y}$, represents the predicted quantity. In this article, we consider regression problems, i.e. $\hat{y}$ is a scalar. Without loss of generality, the methods proposed in this paper apply to vector-valued outputs. Moreover, extensions to binary classification problems would be possible if we consider $\hat{y}$ to be a classification score.

3) THE BLACK-BOX *B*

We assume that we do not know the functional form of B, which is the situation when using an off-the-shelf piece of software. However, it should be noted that our proposal also applies to situations where B may be known but changes in B are not allowed or would be expensive. Therefore, our definition of black box in fact means that B is *immutable* (either because it is unknown, or because it is costly or impossible to replace for any of the given reasons).

B. UNCERTAINTY SCORES OF BLACK-BOXES

We use the term uncertainty score to define any function $\psi(z)$ that can be used to impose an *ordering* of predictions, intuitively in terms of quality. The crucial property of an uncertainty score is that, by thresholding the uncertainty score and keeping only the samples with uncertainty below a given threshold, the error metric of interest or dispersion for that 'refined' subset would be lower than by considering the whole set.

We assume we work with a black-box B and at least have access to the triplets $(z_n, \hat{y}_n, y_n)$, where z are preserved or distorted inputs, $\hat{y}_n$ are estimated predictions and y_n are the ground-truth predictions. In this situation, our goal is to construct a model $\psi(z)$ that provides a quantitative estimate of the uncertainty score for the predicted value $\hat{y}$, given the preserved or distorted inputs z.

C. A NON-BAYESIAN APPROACH

We aim at obtaining uncertainty scores of $\hat{y}$. Broadly speaking, there are two main approaches to elicit the uncertainty of predictions in statistics [15]: either inducing an uncertainty over the model parameters (*epistemic uncertainty*), or fixing the model and assuming the output is noisy *aleatoric uncertainty*. Bayesian approaches are a well-known way of tackling epistemic cases, as they impose a distribution over the parameter space. **Crucially for this work, when dealing with black boxes, one does not have access to the model and thus it is not possible to apply Bayesian approaches**.

Specifically, we focus on the aleatoric component of uncertainty [15], i.e. the variability of the outputs that depends on the input data, since, as we will see next, this can be estimated empirically even if the model is unknown.

IV. METHOD

This section presents the details of different proposed methods for building the uncertainty wrapper function.

The uncertainty wrapper function is defined as an end-to-end differentiable model, $\psi(z)$ that, given a loss function, can be trained to produce an uncertainty score. Taking into account that this function to be approximated may be as or even more complex than the function to be approximated by the black box, we decided to use deep learning models.

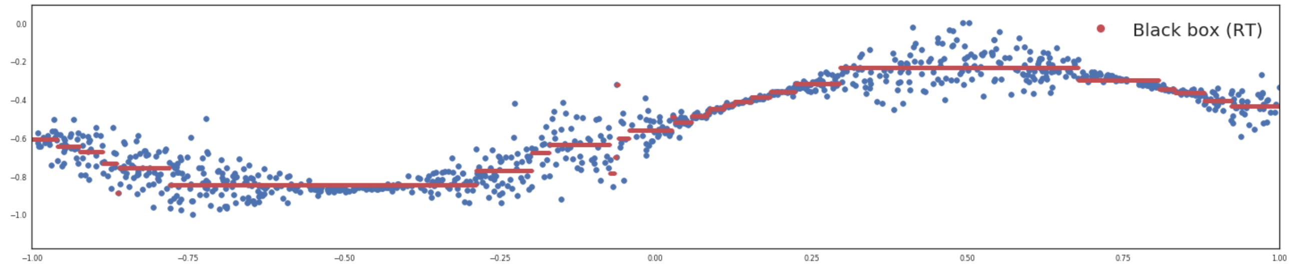
The use of deep learning models have advantages and disadvantages. On the one hand, these models are universal approximators [12] and they are the state-of-the-art methods for solving a wide range of high-dimensional problems [14], [30], [44]. On the other hand, similarly to normal mixtures and Boltzmann machines, deep learning models are not identifiable, which may implies to have different local minima [18]. Given that we are not associating a meaning to each internal weight of the model, this problem is not critical for our scenario. Furthermore, the non-optimal guarantees about the local minima found is not a critical problem in our scenario while this models obtain the state-of-the-art solution for our problem.

More specifically, we have considered three different strategies for obtaining the uncertainty score:

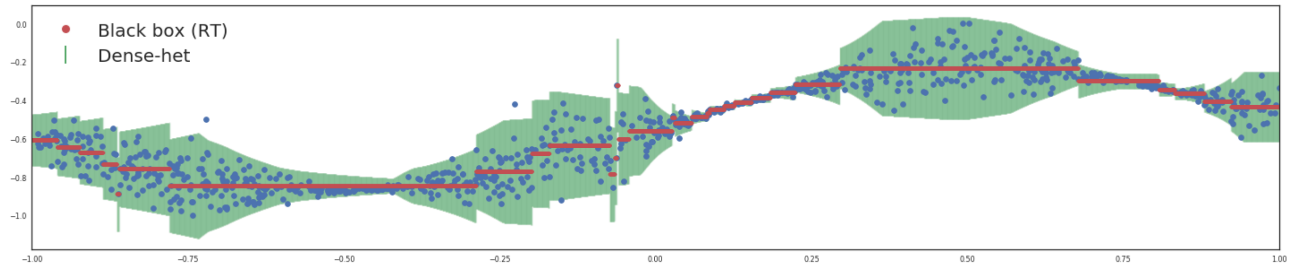
- Modelling the target variable as a parametric distribution and posing the problem of uncertainty estimation as a probabilistic inference problem.
- Estimating the residual error between the prediction $B(x)$ and the target value y.
- Building a quantile regressor to directly estimate 95% and 5% percentiles of the target variable.

The training step consists in optimizing $\psi(z)$ through the use of a specific loss function $\mathcal{L}()$ and a dataset $(z_n, \hat{y}_n, y_n)$.

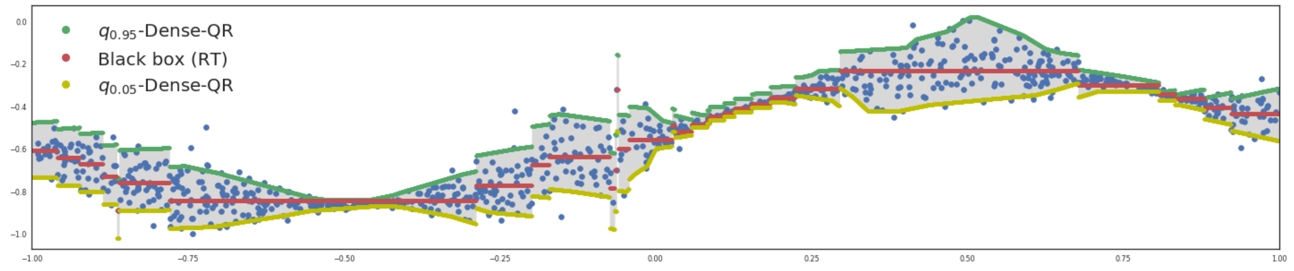
Once $\psi(z)$ has been trained, it can be directly used to predict the uncertainty score of $B(x)$ as $\psi(z)$.



(a) Black-box prediction (based on a Random Tree) over the test points.



(b) Heteroscedastic model approximation based on Eq. (2).



(c) Quantile Regression approximation based on Eq. (4).

FIGURE 2. Uncertainty modelling using different wrappers of a black box estimation based on a regression Random Tree (RT) to approximate a non-linear synthetic data set. Note that the uncertainty score only deviates from the expected values when the black-box forecast is erroneous.

At this point, it is worth emphasizing the difference between the *First* and *Second Order* models. In the *First Order* case, the entry of the uncertainty wrapper, ψ, model will be simply the vector z. However, in the *Second Order* case, the input will comprise the prediction by the black box, B, and the vector z, as follows: $\psi\,(z, B(x))$. Without loss of generality we will continue to use the notation $\psi\,(z)$ to refer to both cases. On the other hand, note that the black-box input continues to be x even in the case of having a *distorted input*, since what is needed to model the *Uncertainty Wrapper, ψ*, is the prediction by the black box, i.e. $B(x)$.

The following sections describe each strategy in more detail.

A. PROBABILITY DISTRIBUTION FITTING

The problem posed is to estimate the target variable distribution $\hat{y} = B(x)$ empirically through observations of triplets $(z_n, \hat{y}_n, y_n)$.

When considering deep learning models, a common choice is to model the target variable distribution as a Normal [6] or Laplacian distribution [9] and solve a Maximum Likelihood estimation problem in order to find the optimal network parameters. Both distributions are a sub-case of the parametric family of symmetric distributions known as Generalized Normal Distribution (GND). Hence, we assume $\hat{y} \sim \mathcal{GN}(B(x), \psi(z), \beta)$, where $\mathcal{GN}$ is a GND with the black-box output, $B(x)$, as a location parameter and the deep learning wrapper, $\psi(z)$, as a scale parameter [28], [41]. Furthermore, β is another parameter to be optimized and, as highlighted earlier, represents the Laplacian distribution when $\beta = 1$ and the Normal distribution when $\beta = 2$. On the whole, the distribution function, shown in Figure 2.b, is:

$$\mathcal{GN}(y \mid B(x), \psi(z), \beta) = \frac{\beta}{2\psi(z)\Gamma(\frac{1}{\beta})} e^{\left(-\frac{|y-B(x)|}{\psi(z)}\right)^{\beta}} \qquad (1)$$

where Γ denotes the Gamma function.

Importantly, the scale parameter of the distribution, $\psi(z)$, is a function that depends on z, the *preserved* or *distorted* information available at the input. Therefore, our goal can be stated as that of finding a functional form of $\psi(z)$ and the parameter β which maximizes the corresponding log-likelihood:

$$\mathcal{L}\left(y, B(x), z\right) = \log \frac{\Gamma(\frac{1}{\beta})}{\beta} + \log \psi(z) + \left(\frac{|y-B(x)|}{\psi(z)}\right)^{\beta} \tag{2}$$

As mentioned earlier, the reason for considering $\psi(z)$ as a deep learning model is clear: we are trying to approximate a function that might be complex and non-linear, meaning a high capacity model seems judicious. Note that estimating the possible variability of an output given the input is a regression problem in itself and therefore as challenging as predicting the expected value. This setting is model-agnostic with respect to the uncertainty wrapper deep learning architecture. Given the deep learning output, $NN(z)$, which by default can take positive and negative values, the only requirement is that $\psi(z)$ needs to always be positive; we therefore apply a `Softplus` function to the output of the deep learning as proposed in other works when scale parameters are predicted [25], [26], [46], i.e. $\psi(z) = \log\left[1 + \exp\left(NN(z)\right)\right]$. In our particular case, however, one crucial detail is that $B(x)$ is fixed, and thus, the loss function will not be optimized with respect to it, so the values of $|y - B(x)|$ are fixed.

Finally, on the one hand we can interpret the loss function of Equation 2 as the sum of a regularization term and a reconstruction term. If $\psi(z)$ is smaller than the scale of these errors, the reconstruction term $\left(\frac{|y-B(x)|}{\psi(z)}\right)^{\beta}$ penalizes the loss value. Otherwise, if $\psi(z)$ is too large, then the regularization term $\log \psi(z)$ becomes dominant. So we have an equilibrium that yields $\psi(z)$ as the predicted scale of the errors depending on the input.

On the other hand, the job of β is less evident, for some fixed set of errors $|y-B(x)|$ and uncertainty scores, $\psi(z)$. The optimal β is such that the shape of the distribution fits better, i.e. large values of β correspond to plateau-like distributions, and low values of β to sting-shaped ones.

B. LIKELIHOOD ESTIMATION OF RESIDUALS

In this case, the aim is to directly estimate the residual error between the black box and the value to be predicted, letting $\psi(z) = |y - \hat{y}|$. Implicitly, we can do this because y is an unknown function of x. Assuming the residuals follow a GND of unknown variance, the loss function for this case is the negative logarithm of the likelihood, out of constants and dependence from the variance:

$$\mathcal{L}\left(y, B(x), z\right) = \log \Gamma(1/\beta) - \log \beta + |y - B(x) - \psi(z)|^{\beta} \tag{3}$$

C. QUANTILE REGRESSION

A well-known approach to compute confidence intervals and deal with prediction with uncertainty is Quantile Regression, in which the target estimate is a certain quantile of the distribution of real values, instead of the mean [19], [27]. The length of the centred 90% confidence interval can be used as a proxy of the uncertainty of the estimation. To do this, we define two functions $\phi^{+}(z)$, $\phi^{-}(z)$ as the $(\frac{19}{20} =)95\%$ and $(\frac{1}{20} =)5\%$ percentiles of the distribution of the residual error, $r = y - B(x)$, for each function and use a surrogate hinge loss to them [39] in the following way:

$$\begin{cases} \mathcal{L}^{+}\left(y, B(x), z\right) = \max\left[19(r - \phi^{+}(z)), \phi^{+}(z) - r\right] \\ \mathcal{L}^{-}\left(y, B(x), z\right) = \max\left[r - \phi^{-}(z), 19(\phi^{-}(z) - r)\right]. \end{cases} \tag{4}$$

The synthetic example presented in Figure 2.c shows how both quantiles are estimated.

As in the cases included in Section IV-A, we define the uncertainty score as the `Softplus` of the quantile difference, $\psi(z) = \log\left[1 + \exp(\phi^{+}(z) - \phi^{-}(z))\right]$.

V. BASELINES UNDER EVALUATION

In order to evaluate our proposals, we compared the previous proposals with two methods that allow us to obtain an uncertainty proxy given a dataset of triplets $\left(z_n, \hat{y}_n, y_n\right)$. Additionally, we defined a simple prediction baseline as a sanity check.

A. NEAREST NEIGHBOUR DISTANCE

The distance to the n-th nearest neighbour in the input space [49], z, can be seen as a proxy of "normality", which is sometimes related to reliability. As a simple baseline, we used the distance to the $5th$ neighbour to sort predictions by reliability. The distance to other neighbour can be considered. We have abbreviated this method as *NN*.

B. NEAREST NEIGHBOUR REGRESSION

As well as using distance, we can also use the targets of the nearest neighbour to obtain a direct estimation of uncertainty [4]. Specifically, if we call $Y_{neigh} = y_{\pi(1)}, \ldots, y_{\pi(K)}$ the targets of the K first neighbours, we use $\text{std}(Y_{neigh})$ as an estimate of the prediction uncertainty. We denote this method as *NN-Reg*.

C. GAUSSIAN PROCESSES

Following the product-of-GP-experts model proposed in [13], we built an ensemble of N Gaussian processes, $\{GP_i\}_{i=1}^{N}$, where each one is trained with a different part of the training set to predict the difference between the black-box prediction and the real value, i.e. $|y - B(x)|$. Thereafter, as each Gaussian process predicts a mean, $\mu_i(z)$, and a variance, $\sigma_i(z)$, one way of defining the uncertainty score could be:

$$\psi(z) = \frac{1}{N} \sum_{i=1}^{N} \mu_i^2(z) + \sigma_i^2(z) \tag{5}$$

In this way, we avoid the scalability problems typically found in Gaussian Processes. We refer to this baseline as *GP*.

D. STANDARD DEVIATION OF z

When forecasting the next value of a time series, one can use the standard deviation of the previous time series points as the simplest means of estimating the uncertainty of the next value. While we would expect this to yield a poor uncertainty proxy, we added it as good experimental practice to make sure that our proposal yields much better results than simple cases. From now on we refer to this as *std*.

VI. EXPERIMENTAL SETTINGS

A. DATA SETS

All the datasets used are of the form $[z, y, \hat{y}_i]$, where $[z, y]$ are the real data, z the available inputs and y the target variable(s), and $\hat{y}_i$ are each of the black-box estimates for y.

1) FORECASTING BANK CUSTOMERS' IMPENDING FINANCIAL EXPENSES AND INCOMES

Following [9], our problem is to forecast upcoming monthly expenses and incomes in a certain aggregated financial category for each bank client. Each time series contains 24 points and the goal is to predict the next aggregated month. To build the dataset, we used 2 million randomly-selected time series for a single-year training set and 1 million more for the test corresponding to the following year.

2) ESTIMATING ELECTRICAL POWER DEMAND

In this problem, we have to forecast the mean electrical power demand in two hours, given the means for each two-hour period over the previous 72 hours. Thus, we have time series of 36 points and the problem is forecasting the next one. The data were prepared from the sets made publicly available by Red Eléctrica of Spain, which can be found at [2]. The public series are at intervals of 10 minutes, so we averaged every 12 points to obtain the mean value for 2 hours. In this experiment, data were captured for the period comprising 1-1-2014 to 18-10-2018, so we have 250,000 points, split evenly between train and test.

3) PREDICTING A BIOLOGICAL RESPONSE CHALLENGE

We also considered a real-world dataset from a public Kaggle challenge [23]. The data comprised $1,776$ numerical descriptors representing the size, shape or elemental constitution of each molecule. The aim here was to predict whether the molecule was seen to elicit a biological response. Although the challenge is a binary classification problem, for the purposes of this article, we regard it as a *regression* task, where it is necessary to predict a real-valued score (0 or 1). The interest of this dataset lies in the fact that one of the participants published the code for her solution [37], which we used as a black box. The dataset had $3,751$ points: 500 were used to train the black box, $2,000$ to train the confidence estimator and $1,251$ were used as a test set.

B. BLACK BOXES USED FOR EVALUATION

The aim of using several black boxes was to simulate different situations encountered in real-world problems, where an interpretation is needed or the function $B(x)$ cannot be changed for practical reasons. For each dataset, we took *existing real systems* as black boxes, and in order to extend the study, in some cases complemented them with additional simulated black boxes, as follows:

For the **Financial dataset**, we used the following black boxes:

- *Mean*: The average forecast of the historical input values: $\hat{y} = \bar{x}$.
- *Last*: The last observed value in the time series with 24 points: $\hat{y} = x_{24}$, which is known as the "naive method" in the forecasting literature [22].
- *In Production*: The in-production system that produces forecasts for the banking app. This system is highly optimized for production purposes, difficult to replace, and uses a number of different models and software components. It outputs a point forecast but not a prediction interval. Therefore, it complies with many of the black-box assumptions described in this work.

For the **Electrical Power Demand dataset**, we considered:

- *Company*: Red Eléctrica's own forecasting consumption, computed by the company itself. These forecasts are available as part of the dataset, but we ignored how the predictive system is designed. Thus, we could be in a distorted input case, following Section III.
- *RT*: A regression tree model with depth 4, available from the `sklearn` library.

For the **Biological Response dataset**, we used a participant's solution to the Kaggle challenge, as published in [37], hereafter referred to as *Kaggle*.

At this point, it is important to highlight the different kinds of black boxes used. Firstly, the initial two black boxes (*Mean* and *Last*) proposed for the Financial dataset are a clearly *preserved input* case, since both the forecasting method and the uncertainty estimation method work with the expense time series data. However, the third case (*In Production*) is a *distorted input* scenario, since the in-production forecaster uses more attributes than just the previously predicted series, which our uncertainty wrapper has no access to. Similarly, the Electrical Power Demand estimation done by the company probably uses additional information that was not available to us (we assume these to be weather conditions, dummies for special dates, etc.), meaning that *Company* is also a *distorted input* scenario.

Finally, we considered *First* and *Second Order Decision* models for all of the alternatives.

C. DEEP LEARNING SPECIFICATIONS

Different architectures were combined with different loss functions. In the two time series datasets, we used recurrent and dense networks (called *LSTM* and *dense*, respectively), but only the latter in the classification/regression dataset.

In the **Financial** and the **Electrical Power Demand datasets**, the dense network has two hidden dense layers with 50 and 20 units, respectively, while the recurrent network has a first hidden layer of 50 LSTM units and a second of 20 dense neurons.

In the **Biological Response dataset**, the dense network has two hidden layers with 3, 000 and 1, 000 units, respectively, due to the high number of input attributes.

The *Second Order* dense *decision models* have the same structure as the *First Order* ones, but adding the black-box output as an extra input, whereas the *Second Order* recurrent *decision models* have an extra dense layer to extract features from the black-box output to 10 units, which feeds the second hidden layer as well as the output of the LSTM layer.

All the deep learning models were implemented using the automatic differentiation library TensorFlow [3] and, specifically, the Keras wrapper [11]. Their corresponding parameters were optimised using a grid search of different parameter combinations for each model. Furthermore, they were trained in two phases using 500 epochs with early stopping and 10% of the training set as a validation set. As explained in Section VI-D, in the first phase, β is trainable so the model learns the shape of the distribution, whereas in the second phase, β is constant, so we have more numerical stability to learn ψ.

Note that when we perform quantile regression, we train two models, one for the interval's upper-bound function and another for the lower-bound one.

On the whole, we considered three different loss functions:

- *het*: Heteroscedastic aleatoric estimation loss, Eq. (2).
- *res*: Deterministic bias estimation loss, Eq. (3).
- *QR*: 90% centered coverage Quantile Regression, Eq. (4).

D. HYPERPARAMETER POLICY OF β

In both frameworks (heteroscedastic and residual), we first optimized the network with all the trainable parameters, and then froze β when it had converged (i.e. low variation of that parameter is regarded as convergence), and continued training the other weights (those used to compute $\psi(z)$). This provides more numerical stability, as minor changes in the value of β affect the loss value.

VII. RESULTS

In this section, we quantitatively and qualitatively evaluate certain properties of the uncertainty wrapper, for both our in-house example and the real-world public datasets.

A. DOES THE UNCERTAINTY WRAPPER PREDICT THE CONFIDENCE?

The first question we wish to solve is whether the uncertainty wrapper has the ability to filter those points where our black box makes larger errors. In other words, do the uncertainty scores rank the predictions by increasing error?

Note that traditional ways of evaluating regression (or forecasting) methods are real-valued error metrics, such as Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE). However, these are computed based only on the predictions $B(z)$ and the true values, whereas here we are looking for a way to evaluate the quality of $\psi(z)$.

1) ERROR-KEEP CURVE

One way to compare the ordering quality of different uncertainty score functions, ψ_k, is by contrasting their different error-keep curves [9]. Figure 3 shows the curves for different methods applied to the financial dataset with respect to the "In Production" industrial black box. Each sub-figure corresponds to a different scoring measure indicated in the corresponding caption. These curve plots, on the y-axis, have the respective cumulative scoring measure, $\mathcal{D}$, corresponding to the subset of the predictions such that $\psi(z) < \kappa$, where κ is a threshold. The x-axis shows the fraction of points under the threshold. Following [16], if $\Psi_i = [\psi_k(z_i) < \kappa]$, the different selected scoring methods are the Mean Absolute Error (MAE),

$$\mathrm{EK}_{MAE}\,(y, z, B(x), \kappa) = \frac{\sum_{i=1}^{N} |y_i - B(x_i)| \cdot \Psi_i}{\sum_{i=1}^{N} \Psi_i}, \tag{6}$$

the Mean Absolute Percentage Error (MAPE),

$$\mathrm{EK}_{MAPE}\,(y, z, B(x), \kappa)) = \frac{\sum_{i=1}^{N} \left|\frac{y_i - B(x_i)}{y_i}\right| \cdot \Psi_i}{\sum_{i=1}^{N} \Psi_i}, \tag{7}$$

the Mean Percentage Error (MPE),

$$\mathrm{EK}_{MPE}\,(y, z, B(x), \kappa)) = \frac{\sum_{i=1}^{N} \frac{y_i - B(x_i)}{y_i} \cdot \Psi_i}{\sum_{i=1}^{N} \Psi_i}, \tag{8}$$

the Root Mean Squared Error (RMSE),

$$\mathrm{EK}_{RMSE}\,(y, z, B(x), \kappa)) = \sqrt{\frac{\sum_{i=1}^{N} (y_i - B(x_i))^2 \cdot \Psi_i}{\sum_{i=1}^{N} \Psi_i}}, \tag{9}$$

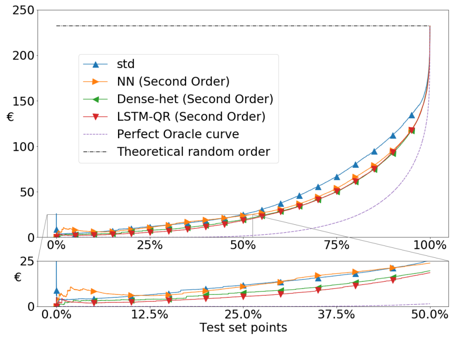
the Root Mean Square Percentage Error (RMSPE),

$$\mathrm{EK}_{RMSPE}\,(y, z, B(x), \kappa)) = \sqrt{\frac{\sum_{i=1}^{N} \left(\frac{y_i - B(x_i)}{y_i}\right)^2 \cdot \Psi_i}{\sum_{i=1}^{N} \Psi_i}}, \tag{10}$$

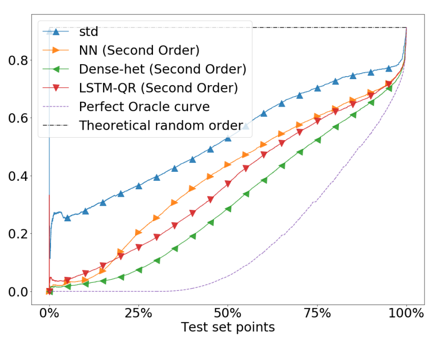
and the Mean Square Percentage Error (MSPE),

$$\mathrm{EK}_{MSPE}\,(y, z, B(x), \kappa)) = \frac{\sum_{i=1}^{N} \left(\frac{y_i - B(x_i)}{y_i}\right)^2 \cdot \Psi_i}{\sum_{i=1}^{N} \Psi_i}. \tag{11}$$

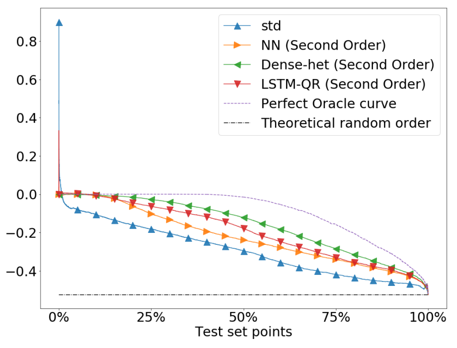
As we can see in Figure 3, for all methods we obtain a trade-off curve showing that the error decreases as we



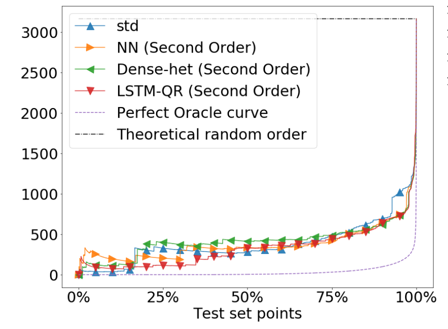
(a) MAE measure, Eq. (6).



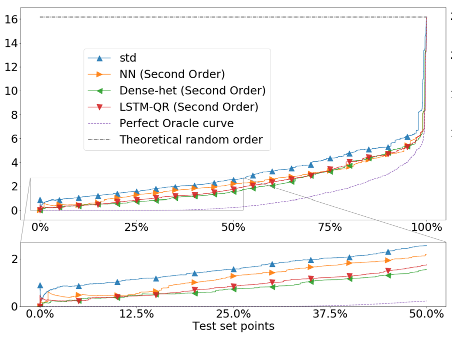
(b) MAPE measure, Eq. (7).



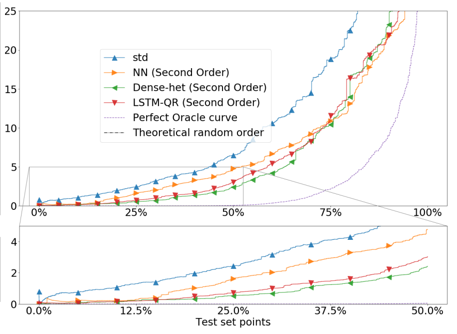
(c) MPE measure, Eq. (8).



(c) RMSE measure, Eq. (9).



(d) RMSPE measure, Eq. (10).



(e) MSPE measure, Eq. (11).

FIGURE 3. Error-keep plot of the *In Production* black box for our Financial Forecasting problem using different scoring measures. Sub-figures (a), (d) and (e) have a zoomed shot of the initial 50% at the bottom.

''accept'' only forecasts with decreasing values of the uncertainty $\psi(z)$. In the experiment presented here, we have successfully added an uncertainty wrapper on top of an industrial black box, and we can now use it to filter the forecasts we are unsure about and not display them to the user.

We are in the scenario that metrics posses advantages in interpretability are preferable due to the predicted value is monetary. Following [40], we consider better to focus on metrics that uses MAE instead of RMSE as they are fundamentally easier to understand than the latter. Consequently, although the standard deviation could seems better, in the initial part, if we only show Figure 3.c case, we can observe that in all other cases the proposed methods based on heteroscedastic networks and quantile regression obtain the best performances.

2) EXHAUSTIVE QUANTITATIVE EVALUATION

Since it is impractical to visualize all error-keep figures for all of the methods, we summarize all of them into a single metric, which we will denote as ordering score. This value will be computed for every combination of dataset, method, black box and order (First and Second order), and evaluated for all possible combinations.

The ordering score quantifies whether the ordering induced by an uncertainty function $\psi(\cdot)$ is close to the perfect or, otherwise, to a random ordering. Its computation details are explained below.

First, let us consider the error-reject curve of a perfect ordering. It is clear that the best possible ordering happens when it is the same as ordering by the real error, that is $\psi_o(z_i) = |B(x_i) - y_i|$. We denote this ideal situation as the ''perfect oracle curve''.

Similarly, the least informed way to sort by uncertainty scores is randomly. Clearly, this would yield a constant error-keep curve with a value corresponding to the MAE of the dataset (up to random fluctuations). We can define

$$\delta_k = \left[\underbrace{\left(\frac{1}{N} \sum_{i=1}^{N} |\psi_k(z_i) - y_i| \right), \ldots}_{\text{Repeat } N \text{ times}} \right] \tag{12}$$

as the vector with the value of the whole MAE of ψ_k.

At this point, where we have defined a lower and upper bound curve, we are able to define the **ordering score** of an uncertainty wrapper function, ψ_k, as

$$\mathcal{S}(\psi_k) = 100 \left(1 - \frac{A(\psi_k) - A(\psi_o)}{\delta_k - A(\psi_o)} \right), \tag{13}$$

where $A(v)$ is the area of the *Error-Keep curve* of the function v. The ordering scores is one minus the ratio between the difference of area between the selected uncertainty wrapper and the oracle divided by the difference of the area of a random ordering criteria and the oracle. Therefore, the closer the ordering score is to 100, the closer it is to a perfect sorting. On the other hand, a value closer to zero (it may even be

TABLE 1. Ordering scores values for each of the methods explained in Section IV and for each of the datasets of Section VI-A.

Dataset	Financial forecasting						Electrical Power Demand				Biology	
Black-box	Mean value		Last value		In Production		Company Forecast		Regression Tree		Kaggle solution	
std	87.1		87.0		86.5		4.8		11.1		-	
NN	73.4	74.0	84.6	88.0	83.5	89.0	5.8 ± 0.0	5.8 ± 0.0	16 ± 0.0	20.5 ± 0.0	25.6 ± 0.0	25.7 ± 0.0
NN-Reg	82.4	84.4	87.9	88.5	88.2	88.6	11.2 ± 0.0	10.8 ± 0.0	15.1 ± 0.0	14.7 ± 0.0	35.6 ± 0.0	35.4 ± 0.0
GP	60.2	60.2	67.7	79.0	64.5	77.9	7.1 ± 0.3	7.5 ± 0.4	25.4 ± 0.3	82.9 ± 0.0	21.6 ± 0.9	21.8 ± 0.8
Dense-res	80.6	81.4	-4.2	5.1	-2.4	3.0	0.6 ± 3.5	1.9 ± 2.2	38.7 ± 3.0	50.4 ± 4.2	4.9 ± 0.0	4.3 ± 1.6
LSTM-res	79.7	78.5	0.9	4.5	0.0	0.0	1.8 ± 7.6	5.4 ± 5.0	42.9 ± 3.8	85.2 ± 1.6	-	-
Dense-QR	90.1	89.9	91.3	91.1	90.7	90.8	18.6 ± 1.4	16.9 ± 1.6	32.1 ± 2.4	18.2 ± 4.8	14.8 ± 4.8	14.1 ± 5.5
LSTM-QR	89.0	88.7	90.8	91.4	90.3	91.0	12.7 ± 0.6	17.1 ± 1.6	32.2 ± 2.9	14.4 ± 5.6	-	-
Dense-het	92.9	91.0	88.8	88.3	88.4	90.6	20.4 ± 0.9	20.8 ± 0.8	50.4 ± 5.8	83.9 ± 4.2	51.6 ± 2.6	52.2 ± 1.9
LSTM-het	93.3	93.4	89.8	91.2	87.8	86.8	17.0 ± 4.9	19.3 ± 3.2	50.4 ± 4.2	71.3 ± 7.7	-	-

Background color meaning: ☐ First Order Decision version of the model ▨ Second Order Decision version of the model

negative given stochasticity) will mean an almost random ordering.

In Table 1 we show the ordering scores values for all the combinations of datasets, black-boxes, methods and First/Second order choice explained in Sections IV, V and VI-A. For all public dataset, the mean and variance ordering scores values of 10 independent executions are reported.

Returning to the original question about detecting whether the uncertainty wrapper improves in our confidence, if we look at the Table 1, we see that all values are positive values, with the exception of some cases corresponding to the *LSTM-res*. Therefore, taking into account the definition of the ordering score, we can ensure that by using the wrapper constitutes an improvement in the performance. Additionally, we can observe that the deep Uncertainty Wrapper strategies proposed in this article are the ones that get the best results for every point compared to other baselines.

Furthermore, in the comparison presented in Table 1 we can observe that the ordering score exhibit relevant variations in scale depending on the complexity of the problem. This complexity is related with the presence of distorted or preserved inputs implying bad performance such as the residual uncertainty wrappers that obtained close to zero or even negative values.

B. WHICH IS THE BEST UNCERTAINTY WRAPPER?

The results of the Table 1 leads us to wonder if there is a method that stands out from the others systematically. While the 1st ranked method varies over the columns of the table, we can see that clearly using the Heteroscedastic methods as wrapper gives us the first or second best position for any problem and black-box. Thus, we can consider that the Heteroscedastic model is the most stable when it comes to getting good generic solutions.

C. FINE-GRAINED ANALYSIS OF ORDERINGS

Table 1 and Figure 3 give insights on the overall *quality* of the orderings induced by uncertainty wrappers.

We would also like to check monotonicity properties of the ordering, i.e. to what degree does the uncertainty score approximately sort by real-error? In other words, we want to match the intuition that "easy to predict" inputs should be assigned low uncertainty scores, while "potentially disastrous prediction" should be avoided.

To that end, we first group the samples by MAE, $|y_i - B(x)_i|$ into 10 bins. These can be interpreted as 10 different degrees of prediction difficulty (from lowest to highest error). When we vary the uncertainty threshold κ we could measure the % of points that fall into each of the 10 bins (y-axis), while the x-axis corresponds to the Keep % (induced by κ). We would expect the low-error bins to capture most % at low Keep rates, and conversely, that the high-error bins dominate at high values of the Keep rate. For instance, this could be used to detect wrong predictions with a high Uncertainty Score and other unwanted scenarios.

In Figure 4, we can show that for all methods displayed, the bins with large error (the purplish ones) appear at the end. On the other hand, we can observe that the behaviour of both Heteroscedastic and quantile regression wrappers is quite smooth even for lower bins (the yellowish ones). Therefore, we can conclude that considering ordering score values is a proper manner to ordering regarding different types of errors.

D. DISCRETIZATION IN LEVELS OF CONFIDENCE

Another important point to verify is the correlation between the ordering induced by each uncertainty wrappers and the 'true ordering' based on the real error (oracle). In this case, for each uncertainty wrapper we could define certain thresholds corresponding to the five required quantiles (i.e. values between the quantiles $0-20$, $20-40$, $40-60$, $60-80$ and $80-100$) and associate them to the five classes, respectively. Note that this is equivalent to defining five 'quality classes' (*Higher Error*, *High Error*, *Medium Error* , *Low error* and *Lower Error*) and computing a sort of confusion matrix between the true classes and the predicted classes.

In Figure 5, we observe the confusion matrix for our *In Production* problem where it is indicated in each box the normalized number of predictions that have coincided between the prediction of the oracle and the chosen uncertainty wrapper.

As we can see in Figure 5, it is easier for all the different presented models to detect the points with higher confidence than the others (given their more yellowish colour). However, the number of the percentage of points contained in

TABLE 2. β **final value after the first part of the optimization for each model and problem.**

Dataset	Financial forecasting						Electrical Power Demand				Biology	
Black-box	Mean value		Last value		In Production		Company Forecast		Regression Tree		Kaggle solution	
Dense-res	0.297	0.299	0.303	0.306	0.304	0.306	2.156 ± 0.000	2.156 ± 0.000	2.210 ± 0.002	2.165 ± 0.001	0.776 ± 0.014	0.777 ± 0.019
LSTM-res	0.298	0.297	0.301	0.304	0.304	0.306	2.155 ± 0.000	2.155 ± 0.000	2.210 ± 0.001	2.168 ± 0.001	-	-
Dense-het	0.629	0.675	0.673	0.673	0.677	0.393	1.170 ± 0.003	1.171 ± 0.004	2.655 ± 0.264	5.706 ± 0.654	0.689 ± 0.001	0.687 ± 0.001
LSTM-het	0.617	0.656	0.189	0.668	0.192	0.201	1.102 ± 0.009	1.190 ± 0.030	3.430 ± 0.414	10.17 ± 1.951	-	-

Background color meaning: ☐ First Order Decision version of the model ▨ Second Order Decision version of the model

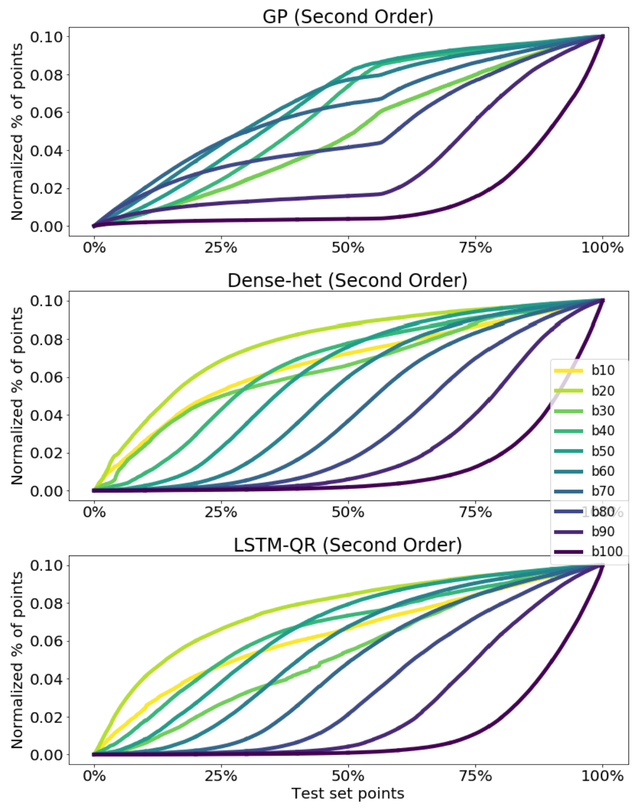


FIGURE 4. Percentage of points of each bin of real MAE error of the *In Production* black-box of our Financial Forecasting problem sorted by the uncertainty wrapper described in the title. Each color corresponds to a different real error bin indicated in the legend.

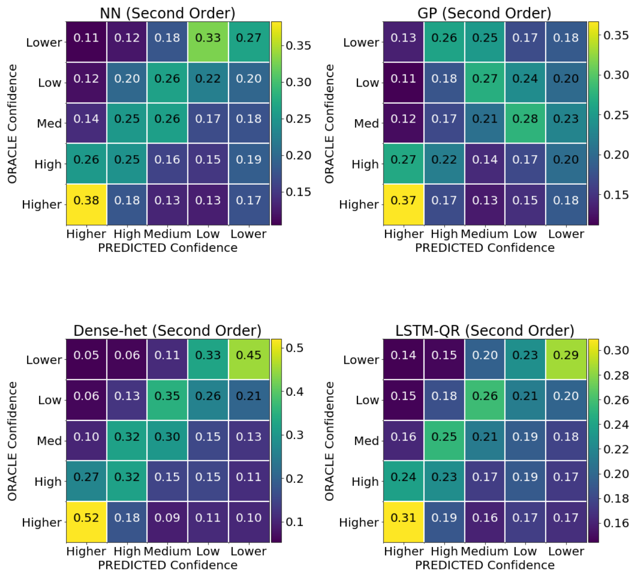


FIGURE 5. Normalized confusion matrix of the problem of classification into 5-levels of confidence for certain models. Each one has its own colour map scale.

such a box on the bottom left is different depending on the model: We can see that all baseline models, as well as the quantile regression model, have a significantly lower value than the *Heteroscedastic* case. This indicates us that, although the ordering score value in our *In Production* problem of the *Heteroscedastic* model may be slightly lower than the quantile regression model, the *Heteroscedastic* model detects in a better way the values that are more reliable. Accordingly, the *Heteroscedastic* model is the one that best orders its *High*, *Medium*, and *Lower* values as well as those predictions that have a high probability of being erroneous in the upper right corner. In short, the model with the most central tendency is the *Heteroscedastic* one.

E. CHECKING THE CONVERGENCE OF β

Before finishing, it is important to analyse the convergence of the extra hyperparameter of the heteroscedastic and estimation of residuals models, β, which we optimize in a first learning phase and allow us to optimize the type of distribution to be fitted, as we explained in Sections IV-A and VI-D. Table 2 shows the final convergence value of the β hyperparameter after the first training phase of all the methods in the different problems. As we can see, the convergence values of β for the problems where the experiment could be repeated converges to a stable value. There is only an exception with the case of the Regression Tree with the *LSTM-het* model where, given that the convergence value of β ended in a high value, the variance also ends up being high. To tackle this situations, a pre-defined value of β could be considered and directly optimize the other parameters of the deep learning wrapper.

F. DISTRIBUTION OF ERRORS REVIEW

Finally, we want to visualize the impact on the probability of an erroneous prediction when the uncertainty score value increases by using the more stable black-box for our Financial forecasting problem: the *Second Order* version of the *Dense-het* model. From the business point of view, this information would be very useful to take the decision on which value of the uncertainty score we discard the predictions. We will now show the whole distribution of the errors sorted by the uncertainty score.

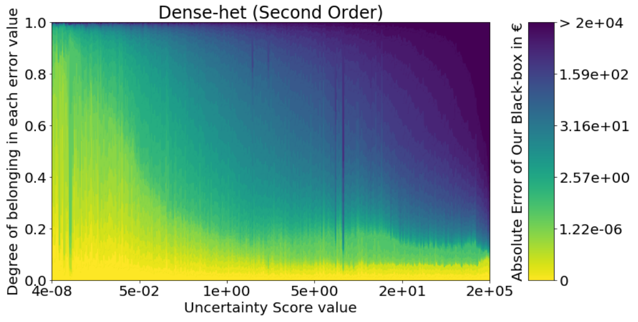


FIGURE 6. Density plot between the uncertainty score value and their corresponding Average Absolute Error produced by the Our *In Production* black-box when it is predicting in the Financial forecasting problem.

1) HOW TO BUILD THE DISTRIBUTION PLOT

The process to generate the Figure 6 is the following: First of all, we stratify into 30 bins the distribution of the absolute values of the error by using our *In Production* black-box for all the test-set, i.e. $|y_i - B(x_i)|$. Then, we assign to each bin a certain color of a defined colormap (as it can be seen in the right of the Figure 6). Afterwards, we order the absolute values of the errors of our black-box prediction by using their respective uncertainty score values. Thereafter, in order to reduce the dimensionality and make the behaviour smoother, we average in groups of 1024 points the previous sorted errors and their corresponding uncertainty score values. Finally, we draw vertically the degree of belonging in each of the error bins for each of the sorted groups.

2) ANALYSIS OF THE DISTRIBUTION RESULTS

Figure 6 exhibits the desired behaviour of a uncertainty score: the left part of the plot (low values of the uncertainty scores) concentrates the samples with lowest error (yellow-ish); the right part of the plot (high uncertainty values) contains a majority of samples with high and very high error (blue-ish).

These trade-off plots can also be used to inspect or debug the failure cases. For instance, we observe a yellow-ish band (bottom of the plot) with high uncertainty scores, which corresponds to samples with low error that the uncertainty model missed. Also, we observe some spikes, which may correspond to specific patterns which do not follow the expected trend. Again, it would valuable to inspect these cases, but we recall the global error-keep trend of the plot remains as desired.

VIII. CONCLUSIONS

Nowadays, automatic predictive systems are more and more commonly used to tackle real-world problems. This implies that increasingly these systems make decisions in problems where the cost of an erroneous prediction is higher than the reward of a correct one, in which obtaining a global good accuracy is not enough. Consequently, the uncertainty regarding this predictive process must be modelled and pointwise predictive system must be replaced by models that considers their uncertainty too. Specifically, we tackle the situation where the cost of replacing an in-production model is not advisable but the uncertainty modelling is required.

In this paper, we propose adding an aleatoric uncertainty wrapper on top of any in-production pointwise predictive system considered as a black-box, i.e. no-assumptions about their internal structure is done and, therefore, it can be a non-parametric predictive systems or even a handcrafted rule-based systems. This freedom ensures that any beneficial property of the original predictive system is preserved although this implies that not all types of wrapper models and uncertainties can be considered.

The proposed methods were applied in different real-world problems using several black-boxes. The most important are the uncertainty modelling of a real electrical power demand forecaster that is currently in-use by the electrical company, the uncertainty modelling of a publicly available solution for a biological response regression and, finally, the black-box uncertainty prediction over the financial expenses and incomes of the bank users that we have currently in-production.

From the whole range of different alternatives proposed in the article, we can see that reports better performance is the second order heteroscedastic model based on a Generalized Normal Distribution fitting. Based on this analysis, we show an improvement in terms of detecting the degree of confidence and, therefore, apply an appropriate policy for each case. Additionally, several ways to visualize the results and to help this decision are proposed.

To sum up, this work is a further step in the research to improve the reliability and robustness of currently in-production systems and state-of-the-art models. Future work could be continued in lines such as capturing other types of uncertainty, as well as, extending the proposed methods for classification problems.

REFERENCES

[1] (Jun. 11, 2020). *Amazon Sagemaker Developer Guide*. [Online]. Available: https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html

[2] (Jun. 11, 2020). *Red Eléctrica: Real-Time Demand and Generation*. [Online]. Available: https://demanda.ree.es/visiona/peninsula/demanda/total

[3] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.

[4] N. S. Altman, "An introduction to kernel and nearest-neighbor non-parametric regression," *Amer. Statistician*, vol. 46, no. 3, pp. 175–185, Aug. 1992.

[5] C. M. Anderson-Cook, "Generalized additive models: An introduction with R. Simon N. Wood," *J. Amer. Stat. Assoc.*, vol. 102, pp. 760–761, 2007.

[6] C. M. Bishop, *Pattern Recognition and Machine Learning* (Information Science and Statistics). Berlin, Germany: Springer-Verlag, 2006.

[7] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," 2015, *arXiv:1505.05424*. [Online]. Available: http://arxiv.org/abs/1505.05424

[8] A. Brando, J. A. Rodriguez, J. Vitria, and A. R. Muñoz, "Modelling heterogeneous distributions with an uncountable mixture of asymmetric laplacians," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8836–8846.

[9] A. Brando, J. A. Rodríguez-Serrano, M. Ciprian, R. Maestre, and J. Vitrià, "Uncertainty modelling in deep networks: Forecasting short and noisy series," in *Proc. ECML*, 2018, pp. 325–340.

[10] S. Chan, T. Stone, K. P. Szeto, and K. H. Chan, "PredictionIO: A distributed machine learning server for practical software development," in *Proc. 22nd ACM Int. Conf. Conf. Inf. Knowl. Manage. CIKM*, 2013, pp. 2493–2496.

[11] F. Chollet. (2015). *Keras*. [Online]. Available: https://keras.io

[12] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control, Signals, Syst.*, vol. 2, no. 4, pp. 303–314, Dec. 1989.

[13] M. Deisenroth and J. W. Ng, "Distributed Gaussian processes," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1481–1490.

[14] L. Deng and D. Yu, "Deep learning: Methods and applications," *Found. Trends Signal Process.*, vol. 7, nos. 3–4, pp. 197–387, Jun. 2014.

[15] A. D. Kiureghian and O. Ditlevsen, "Aleatory or epistemic? Does it matter?" *Struct. Saf.*, vol. 31, no. 2, pp. 105–112, Mar. 2009.

[16] T. Fomby, "Scoring measures for prediction problems," Ph.D. dissertation, Dept. Econ., Southern Methodist Univ., Dallas, TX, USA, 2008.

[17] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1050–1059.

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge, MA, USA: MIT Press, 2016.

[19] L. Hao, D. Q. Naiman, and D. Q. Naiman, *Quantile Regression*, vol. 149. Newbury Park, CA, USA: Sage, 2007.

[20] F. E. Harrell, *Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis*. Cham, Switzerland: Springer, 2015.

[21] J. M. Hernández-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of Bayesian neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1861–1869.

[22] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*. Melbourne, VIC, Australia: OTexts, 2014, Sec. 2.3.

[23] B. Ingelheim. (2012). *Predicting a Biological Response*. Kaggle Challenge. Accessed: Jul. 2, 2020. [Online]. Available: https://www.kaggle.com/c/bioresponse/data

[24] K. S. Kasiviswanathan and K. P. Sudheer, "Quantification of the predictive uncertainty of artificial neural network based river flow forecast models," *Stochastic Environ. Res. Risk Assessment*, vol. 27, no. 1, pp. 137–146, Jan. 2013.

[25] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5574–5584.

[26] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3581–3589.

[27] R. Koenker and K. Hallock, "Quantile regression," *J. Econ. Perspect.*, vol. 15, no. 4, pp. 143–156, 2001.

[28] T. Koski, "Scale parameter," in *Sf 2955: Computer Intensive Methods*. KTH Royal Institute of Technology, 2019.

[29] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6402–6413.

[30] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[31] J. Liu, J. Paisley, M.-A. Kioumourtzoglou, and B. Coull, "Accurate uncertainty estimation and decomposition in ensemble learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8950–8961.

[32] A. Malinin and M. Gales, "Predictive uncertainty estimation via prior networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7047–7058.

[33] J. Mena, A. Brando, O. Pujol, and J. Vitrià, "Uncertainty estimation for black-box classification models: A use case for sentiment analysis," in *Proc. Iberian Conf. Pattern Recognit. Image Anal.* Cham, Switzerland: Springer, Jul. 2019, pp. 29–40.

[34] J. Mena, O. Pujol, and J. Vitria, "Uncertainty-based rejection wrappers for black-box classifiers," *IEEE Access*, vol. 8, pp. 101721–101746, 2020.

[35] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar, "MLlib: Machine learning in Apache Spark," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1235–1241, 2016.

[36] S. J. Oh, B. Schiele, and M. Fritz, "Towards reverse-engineering black-box neural networks," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Cham, Switzerland: Springer, 2019, pp. 121–144.

[37] E. Olivetti. (2012). *Kaggle_Pbr*. [Online]. Available: https://github.com/emanuele/kaggle_pbr

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and J. Vanderplas, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[39] F. C. Pereira, C. Antoniou, J. A. Fargas, and M. Ben-Akiva, "A metamodel for estimating error bounds in real-time traffic prediction systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 3, pp. 1310–1322, Jun. 2014.

[40] R. G. Pontius, O. Thontteh, and H. Chen, "Components of information for multiple resolution comparison between maps that share a real variable," *Environ. Ecological Statist.*, vol. 15, no. 2, pp. 111–142, Jun. 2008.

[41] A. V. Prokhorov, "Scale parameter," in *Encyclopedia of Mathematics*. Springer, 2019.

[42] C. E. Rasmussen, "A practical Monte Carlo implementation of Bayesian learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 1996, pp. 598–604.

[43] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Berlin, Germany: Springer, Feb. 2003, pp. 63–71.

[44] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[45] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," in *Proc. NIPS*, 2015, pp. 2503–2511.

[46] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther, "Ladder variational autoencoders," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3738–3746.

[47] N. Tagasovska and D. Lopez-Paz, "Single-model uncertainties for deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 6414–6425.

[48] M. Teye, H. Azizpour, and K. Smith, "Bayesian uncertainty estimation for batch normalized deep networks," 2018, *arXiv:1802.06455*. [Online]. Available: http://arxiv.org/abs/1802.06455

[49] H. R. Thompson, "Distribution of distance to nth neighbour in a population of randomly distributed individuals," *Ecology*, vol. 37, no. 2, pp. 391–394, Apr. 1956.

[50] B. Ustun and C. Rudin, "Supersparse linear integer models for optimized medical scoring systems," *Mach. Learn.*, vol. 102, no. 3, pp. 349–391, Mar. 2016.

AXEL BRANDO received the master's degree in artificial intelligence and two degrees in mathematics and computer science. He is currently pursuing the Ph.D. degree in industrial with the Universitat de Barcelona and BBVA Data and Analytics. His research interest includes uncertainty modeling by using deep learning with papers in conferences, such as the Neural Information Processing Systems (NeurIPS) and the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD) Conference.

DAMIÀ TORRES worked as an intern researcher at BBVA Data and Analytics. Currently, he is an Advanced Mathematics master's degree student at Universitat of Barcelona. He has taken part in research projects in pure mathematics at Newton Institute, Cambridge, and Centre de Recerca Matemàtica, Barcelona. In addition, he holds many student awards, among them four international first prizes in university mathematics competitions.

JOSE A. RODRÍGUEZ-SERRANO received the Ph.D. degree in computer vision from the Universitat Autonoma de Barcelona. Since 2015, he has been a Senior Data Scientist with BBVA Data and Analytics. He has 12 years of research experience, mostly in industry. He has authored top-tier journal and A* conference papers. He holds over 20 patents.

JORDI VITRIÀ received the Ph.D. degree, in 1990. He has directed 12 Ph.D. theses in the area of machine learning and computer vision.

He was a Researcher with the Computer Vision Center, Universitat Autònoma de Barcelona. Since 1990, he has been the Director of the Laboratory of Electrotechnical Materials, PUB. From 1990 to 2007, he was an Associate Professor with the Universitat Autònoma de Barcelona. From 1995 to 1999, he was the Director of the Electromechanical Energy Conversion Equipment Research Center. Since 2007, he has been the Vice Dean of the Faculty of Electrical Engineering, PUB. He joined the Department of Mathematics and Computer Science, Universitat de Barcelona, in 2007, where he is currently a Professor of algorithmics and data science and the Head of the Research Laboratory on Deep Learning and Applications. Since 2007, he has been a Professor of computer science with the Universitat de Barcelona. He is currently a Professor with the Department of Electrical Machines and Materials, PUB. He has authored more than 100 peer-reviewed articles. He holds eight international patents. He has published more than 200 scientific articles and books in the field of materials for electrical engineering and insulation systems. His research interests include aging mechanisms of electrical insulation, insulation systems testing, polymers breakdown, electrical and water treeing, and electrical materials (dielectrics and composites). He is a member of CIGRE and an Observer of CIGRE D2 SC.