Multi-output Regression for Imbalanced Data Stream

Tao Peng, Sana Sellami, Omar Boucelma, Richard Chbeir

Multi-output Regression for Imbalanced Data Stream

Tao Peng[1], Sana Sellami[1], Omar Boucelma[1] and Richard Chbeir[2]

[1]*Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France*
[2]*Univ Pau & Pays Adour, E2S-UPPA, LIUPPA, EA3000, Anglet, France*
Email: tao.peng@univ-amu.fr, sana.sellami@univ-amu.fr, omar.boucelma@univ-amu.fr, richard.chbeir@univ-pau.fr

In this paper we describe an imbalanced regression method for making predictions over imbalanced data streams. We present MORSTS (Multiple Output Regression for Streaming Time Series), an online ensemble regressors devoted to non-stationary and imbalanced data streams. MORSTS relies on several multiple output regressor submodels, adopts a cost sensitive weighting technique for dealing with imbalanced datasets, and handles overfitting by means of the K-fold cross validation. For assessment purposes, experiments have been conducted on known real datasets and compared with known base regression techniques.

Keywords: Data streams; imbalanced dataset; time series; multi-output regression; non-stationarity; overfitting

Received ; revised

1. INTRODUCTION

In this paper, we are interested in the multi-output regression problem in imbalanced data stream. Multi-output regression aims to predict multiple outputs given an input [1]. In a continuous setting (regression), according to [2], imbalanced regression faces two main challenges: (1) "provide a principled approach capable of describing non-uniform preferences over continuous domains", and (2) "finding appropriate evaluation and optimization criteria capable of improving the predictive ability of models towards extreme values".

As a use case of a continuous imbalanced setting, let's consider the imbalanced smart meter data example [3]. The problem we address here is forecasting the consumption of electrical appliances, a central problem in power management systems for Smart Homes [4]. Appliances can report their energy consumption at specific time intervals via a smart plug.

In this context, data may be *correlated* and *imbalanced*. Correlation results from devices that have correlated consumption, and imbalance may express the presence of extreme / rare values.

To address imbalanced data learning issues, several strategies have been proposed [5]: 1) Under-/Over-sampling, 2) Cost sensitivity strategy, and 3) Online ensemble learning which combines the two previous strategies, in case of a data stream. The under-/over-sampling consists of either adding synthetic instances of the minority classes in the classification problem, such as SMOTE [6], or deleting instances of a majority class, such as NearMiss [7], to obtain a re-balanced dataset. The cost sensitivity strategy means that some data instances (e.g. wrongly predicted data) have higher weights in their loss function calculation. The assumption behind this is that data instances in sparse intervals are easily ignored in the learning process. Therefore, the weight of incorrectly predicted instances must be increased to make the learning process more biased towards incorrectly predicted instances (which tend to belong to sparse intervals) [5]. An online ensemble regression model is a set of individual regression models (submodels) whose predictions are combined to predict the labels of newly entered instances in real time. According to [9, 8], online ensemble regression with cost-sensitive strategy is a promising approach for dealing with imbalanced data as it can provide stable prediction accuracy in imbalanced data streams.

In our work, we adopt a multi-output regression formalism. Most of the existing multi-output regression algorithms [10] operate on a batch mode and adopt the independent and identically distributed (i.i.d) data assumption. Conversely, streaming regression relies on sequential data ingestion, makes predictions for a limited time and may deal with concept drift. Therefore, we believe that multi-output streaming regression is suitable for predicting energy consumption of several appliances.

There exist two kinds of multi-output regression methods [1]: *local methods* that transform the multi-

output problem into single-output problems and *global methods* (aka *adaptation method*) that adapt single-output ones to handle multi-output datasets. Also, authors [1] claim that *"an adaption method has several advantages over problem transformation methods: it is easier to interpret a single multi-output model than many single-target models and it ensures better predictive performance especially when the targets are correlated."* In addition, there is a risk of **overfitting** [11], compared to the similar single-output model, because a model with multiple outputs have more parameters to determine than its version with single output.

In this paper, we propose MORSTS, a Multiple Output Regression method for imbalanced data stream. MORSTS is an *algorithm adaptation method*, with the following characteristics: 1) all submodels are multiple output (e.g., LSTM, Decision Tree), 2) it handles imbalanced data thanks to a cost sensitive strategy, and 3) it provides an overfitting mitigation mechanism by means of k-fold cross validation (KFCV). Experiments on two real datasets demonstrate the efficiency of MORSTS compared to other models.

This paper is organized as follows: Section 2 presents a formalization of the problem. We present a literature review in section 3. Section 4 describes the proposed method and section 5 presents the experiments performed on a real data set. Finally, we conclude this paper in section 6.

2. PROBLEM STATEMENT

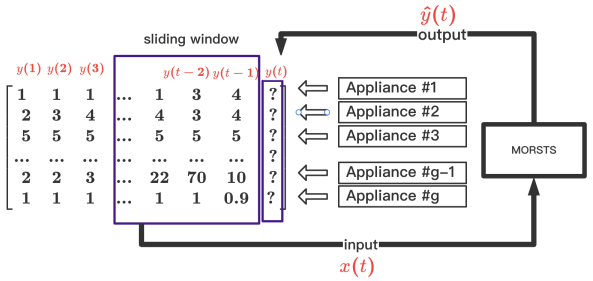
Recall that, in our use case scenario, we are dealing with data (values) that are continuously emitted by sensors. As described in [12], we have to accommodate two concepts: data streams and time series. Hence, to better cope with the situation, we define the concept of Streaming Time Series (definition 2.1) in adapting the definition of *Time Series Stream* provided in [12].

DEFINITION 2.1. *[Streaming Time Series (STS)] is a continuous (unbounded) flow of input data where each instance is a vector of real values:* $STS = [y(1), y(2), ..., y(t-1), y(t)]$ *, where* $y(i) \in \mathbb{R}^g, \forall i \in [1, 2, ..., t-1, t]$*, t is the timestamp of the last value entered, t increases over time and g is the length of the vector that is greater than or equal to 1.*

Fig. 1 illustrates an excerpt of energy consumption of some appliances at different points in time. Each column represents a vector of g values, where g is the number of appliances. The explained variable of the model is the energy consumption of all appliances at a time t. For example, $y(t-1)$ is a vector of length g, representing the energy consumption of g appliances at time $t-1$. As described in definition 2.2, at time $(t-1)$, the multi-output regression model takes $x(t)$ as input, and makes a prediction for $y(t)$. In order to predict the consumption for the next time interval, we use a sliding window (noted as $x(t) = [y(t-WSW), ..., y(t-2), y(t-1)]$, where WSW is the size of the sliding window) that captures the recent historical consumption values of all g appliances. The sliding window size can be adjusted depending on specific scenarios/dataset.

DEFINITION 2.2. *[Multi-output regression problem] Let* $\{y(1), ..., y(t-1), y(t))\}$ *an output vector of length g, representing the energy consumption of g appliances at time t and* $\{x(1), ..., x(t-1), x(t))\}$ *an input vector of length g', representing the recent historical energy consumption of g appliances before time t and depending on two factors: the size of the sliding window (noted WSW) and g which is the number of appliances. Then, $g' = WSW * g$. At time $(t-1)$, the multi-output regression model takes $x(t)$ as input, and makes a prediction for $y(t)$ as output (Fig. 1). The task is to learn a multi-output regression model consisting of finding a function H that assigns to each instance, given by the input vector x, an output vector y of g target values.*

FIGURE 1. Home appliance energy consumption data and MORSTS inputs and outputs presented in Streaming Time Series (STS) format.



3. RELATED WORK

Because MORSTS deals with both multiple output regression and imbalanced data streams, this section reviews some work that are relevant to both aspects.

3.1. Regression in imbalanced data stream

In [13], authors proposed an On-line Weighted Ensemble (OWE) which is a set of single output regressor models in the non-stationary data stream and uses a cost-sensitive boosting strategy to assign small errors to the models that predict accurately the samples predicted poorly by the ensemble.

Experiments performed on several real-world datasets show that OWE handles concept drift and outperforms online ensemble methods such as ILLSA [14] or AddExp [15], which do not have the cost sensitive strategy.

Learn.++R2C [16] is a framework that transforms a single-output regression problem into a single-output classification problem, and then uses Learn.++NSE

[17] a well-known online ensemble classification algorithm with a built-in cost-sensitive strategy to solve this single-output classification problem. Experiments with a real traffic dataset, showed that Learn.++R2C achieves better prediction accuracy results than SARIMA, Passive Aggressive Regression and Regression Tree. However, these two methods are single output variable prediction.

In [18], authors propose two sampling strategies based on Chebyshev's inequality and in the relevance of rare extreme values to train learning models over a balanced data stream once the incoming data stream is imbalanced. This work is based on the hypothesis that the rare cases have extreme target values. Experiments performed on fourteen benchmark data sets showed a reduction of the prediction errors over the rare cases.

3.2. Multi-output Regression in Data Streams

In [19], authors propose an incremental model trees for multi-target regression (FIMT-MT) that relies on the principles of tree-based predictive clustering and the probability boundary analysis (PBA) to partition decisions. Linear models are computed for each target separately, by incremental training of perceptrons. Experiments conducted on both stationary synthetic and real data show that this approach is more accurate and requires less memory compared to a set of single output regression trees.

ILLSA [14] (Incremental Local Learning Soft Sensing Algorithm) is a set of RPLSs (Recursive Partial Least Squares) for multiple output tasks in a data stream. RPLS is an adaptation multi-output regressor that can be updated incrementally. ILLSA uses a two-step phase: 1) in the initial phase, data is divided into samples containing different concepts (via T-test) and RPLS submodels are trained separately; 2) in the online phase, for each incoming data item, the RPLSs' weights are first adjusted using a posteriori probabilities in a Bayesian network, and then the RPLSs are updated with the incoming data. ILLSA does not add or remove submodels, which makes its computation efficient. However, if in the initialization data does not cover the concepts (distributions) in the data stream, the overlooked concepts will affect the accuracy of the prediction. Experiments based on real sensor data show that ILLSA outperforms individual RPLS. In [20], iS-PLS, a model based on RPLS is proposed for multiple output prediction in the data stream. iS-PLS aims to find a low-dimensional subspace to maximize the correlation between inputs and outputs. Experiments with financial data demonstrate the improved accuracy of iS-PLS over RPLS.

In [21], authors proposed an online multiple-output regression method for stream data called MORES which learns the structures of the regression coefficients change and the residual errors in order to refine the model. In MORES, the input vector is multiplied by a learned regression coefficient matrix to obtain the output vector. The matrix of coefficients is updated progressively while respecting certain constraints: 1) the new matrix should not be too different from the old one and 2) the prediction error for the new instances under the new matrix should not be too large. Experiments based on one synthetic and three real data sets show that MORES outperforms iS-PLS in terms of efficiency and accuracy.

In [22], authors presented a rule-based algorithm called AMRules-S. AMRules-S decomposes the multi-output problem into several multi-output sub-problems by learning Adaptive Model Rules (e.g. Decision Tree) and the correlated output variables are placed in the same multi-output sub-problems (e.g. leafs in Decision Tree). The goal is to highlight the correlation between the output variables in each multi-output sub-problem and an adaptation multi outputs MLP (Multilayer Perceptron) makes prediction. Experiments with simulated and real data sets show that AMRules-S is more efficient than individual MLP.

An online multi-output regression system called MORStreaming (Multi-Output Regression System for Streaming Data) was proposed in [10] for solving multiple-output regression problem of streaming data. MORStreaming used an incremental topology learning to select a sub-set of the historical data (i.e. sampling) to represent the current data distribution. So, each multi-output sub-problem has one multi-output submodel which corresponds to a subset of the historical data and an instance-based prediction model (similar to KNN). Experimental results on simulated and real data show that MORStreaming is more accurate than the variants of AMRules-S and variants of Trees.

Note that works described above do not tackle data imbalance. Indeed, learning imbalanced data stream is a challenging task and most of the existing works rely on binary and multi-class classification in imbalanced data streams, as claimed in [23], where few of them addressed regression with imbalanced data due to its complexity [5, 24, 2]. In particular, for multiple output regression, it is possible that some output variables involve data imbalance while others do not. In this case, applying the Under-sampling /Over-sampling according to a few variables that are involved in data imbalance may result in creating a new data imbalance on other variables.

In addition, in a non-stationary imbalanced data stream, the cost-sensitive strategy should be used in conjunction with online ensemble learning to ensure better accuracy [9, 8].

4. MORSTS: MULTIPLE OUTPUT REGRESSION ALGORITHM

In this section, we describe MORSTS (Fig. 2), a Multiple Output Regression method for Streaming Time Series which has the following characteristics:

1. MORSTS adopts an *Ensemble* approach, with

several (multiple output regression) submodels such as LSTM and decision trees. A final prediction is obtained with a weighted vote of all submodels.
2. MORSTS is an online model, and its predictions are in real time. The weights of the submodels are updated per batch according to their prediction error in the current batch. In each batch, the submodel with the lowest weight is replaced by a new submodel.
3. It relies on a cost sensitive strategy for handling data imbalance: a weight of each data instance is calculated according to its prediction error.
4. MORSTS uses k-fold cross validation (KFCV) to handle overfitting. For each new submodel, KFCV error must be below a threshold which is positively correlated with a voted prediction error.

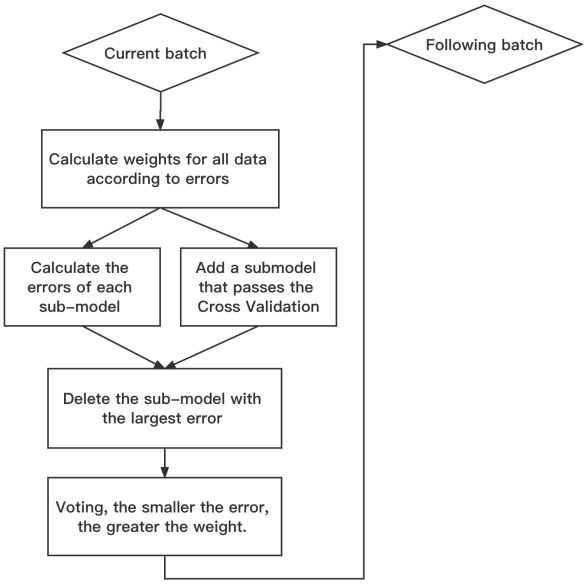
4.1. Method description

We present a formal description of MORSTS.

- if $(x(i), y(i))$ is an instance of the training data at time i, $x(i) \in \mathbb{R}^{g'}$ is a vector describing independent variables,
- $y(i) \in \mathbb{R}^g$ is the vector of dependent variables,
- the multi-output regression function H can be defined as

$$\begin{aligned} H(x(i)) &\rightarrow \hat{y}(i) \\ x(i) &\in \mathbb{R}^{g'} \\ \hat{y}(i) \text{ is an estimation of } y(i) &\in \mathbb{R}^g \end{aligned} \tag{1}$$

FIGURE 2. Brief description of the process and characteristics of MORSTS



4.2. Algorithm

Algorithm 1 illustrates the pseudo-code description of MORSTS (resp. the notations). Notations are detailed in Table is1.

Algorithm description: **If the current batch is the first (line 2)**, a new submodel h^1 is trained with $\mathcal{D}^1$, according to an algorithm randomly chosen in the set of algorithms (lines 3 and 4). This new submodel is added to the set of submodels (line 5). Because this is the first submodel, the current hypothesis is equivalent to this new submodel (line 6).

If the current batch is not the first (line 7), the predictions in the t*th* batch are given according to the current hypothesis H^{t-1} (line 8). The errors of predictions are also computed (line 9). The weights of all instances are assigned according to their prediction error (line 10). Then, for all submodels (line 11), their average of weighted error for the current batch is computed (lines 12, 13, 14).

In each batch, a new submodel will be added. An algorithm is randomly chosen (line 19) and the KFCV method is used to predict all instances in the current batch (line 20). The error of each instance of the current batch (line 21) is calculated, noted $err^t(i)$. We obtain the average of the weighted prediction errors, noted $\varepsilon^t_{h^t}$ (line 22). If $\varepsilon^t_{h^t}$ satisfies the condition, i.e. $\varepsilon^t_{h^t} < \varepsilon^t_{H^{t-1}} \cdot V * *(r-1)$ (line 23), a submodel is trained according to the chosen algorithm using all the instances of the current batch (line 24). This new submodel is added to the set of submodels (line 25). The weight of this new model is based on its KFCV error (weighted with the data weights). Rather, if the above condition is not satisfied (i.e. $\varepsilon^t_{h^t} >= \varepsilon^t_{H^{t-1}} \cdot V * *(r-1)$), a new attempt is made (line 27), and r increases by 1 (line 28).

If the number of submodels exceeds a predefined maximum number, then the submodel with the largest weighted prediction error is removed (lines 31 and 32).

The vote weight of each submodel is inversely proportional to the (weighted) average prediction error of the current batch and inversely proportional to the age (i.e. length of time from training to current) of the submodel (lines 34 and 35). At the end, according to the weights of the vote of the submodels, all the submodels vote to generate the hypothesis, noted H^t (line 37).

Algorithm complexity: The theoretical temporal complexity of the ensemble regressor/model is positively related to the number and complexity of the submodels [14, 16, 10]. Since MORSTS relies on several multiple regressor submodels, the complexity of these collective models remains a challenge [15, 17, 22] because of the non-uniformity of the complexity of the submodels. In the worst case, the time complexity of one submodel predicting N vectors is denoted as C_{pred}, the time complexity of updating a model with N vectors is denoted as C_{update}, the complexity correspond-

TABLE 1. MORSTS Algorithm Notations

Notation	Explanation
	All superscripts and subscripts are indices
$**$	exponentiation
t	the number/index of batch, $t = 1, 2, ...$
$\mathcal{D}^t$	tth batch
m^t	the size of $\mathcal{D}^t$ (number of data instances in tth batch)
i	the number/index of instances in each batch, $i = 1, ..., m^t$
EAR	a set of multiple output regression algorithms
$algo$	a multiple output regression algorithm
ESM	the set of submodels
h^t (resp. h^j)	the submodel that is trained with tth (resp. jth) batch
T	the maximum number of submodels
k	the parameter of KFCV
g'	the length of the vector as inputs
g	the length of the vector as outputs
$\left(x^t(i); y^t(i)\right)$	the ith instance (tuple) in tth batch, and $y^t(i) \in \mathbb{R}^g$
$x^t(i)$	the explanatory variables of the ith instance in the tth batch
$y^t(i)$	the explained variables of the ith instance in the tth batch
$y_f^t(i)$	the fth dependent variable of $y^t(i)$, $1 \leqslant f \leqslant g$
a and b	the slope and inflection point of sigmoids
H^t	the hypothesis of MORSTS in the tth batch
$\left[\hat{y}_1^t(i), .., \hat{y}_g^t(i)\right]$	the estimation of $y^t(i) = \left[y_1^t(i), .., y_g^t(i)\right]$
$err^t(i)$	the prediction error of ith instance in tth batch
$\Theta^t(i)$	the weight of ith instance in tth batch
$\varepsilon^t_{H^{t-1}}$	the average of weighted error of H^{t-1} in tth batch
V	the validation parameter of the submodel, $V > 1$
r	the number of attempts to generate a validated submodel in the current batch
$\varepsilon^t_{h^j}$	the average error (weighted according to Θ^t) of h^j in the tth batch
$W^t_{h^j}$	vote weight of the h^j in the tth batch

ing to initializing a submodel with these N vectors is denoted as C_{ini}. In the worst case, it takes N_{try} attempts to pass the k-fold validation test. Therefore, we can express the time complexity MORSTS as follows: in order to predict N vectors, the time complexity is $M * (C_{pred} + C_{update}) + N_{try} * k * C_{ini}$ where M is the number of submodels.

5. EXPERIMENTS

In this section, we present the experiments that have been conducted on two real-world datasets in order to assess its accuracy and performance. Source code is available at the following Git repository: `https://github.com/jy02407380/MORSTST`.

5.1. Description of the datasets

Our experiments are performed on two datasets: UK-DALE [3] and CU-BEMS [4].

- UK-DALE is derived from the electricity consumption of appliances in five UK households with one minute intervals. We used the "House 1 of UK-DALE" data from the energy consumption of 48 household appliances, over the period November 2013 to January 2016. A sample of this data is shown in table 2. Fig. 3-(a) and Fig. 4-(a) illustrate respectively the imbalance and the non-stationarity of UK-DALE dataset.
- CU-BEMS is a smart building energy data set in Bangkok, Thailand [5]. In our experiments, we used the energy consumption of 22 devices installed in an office over the period from January 2019

[3] Domestic Appliance-Level Electricity dataset available here: https://jack-kelly.com/data/

[4] https://www.kaggle.com/datasets/claytonmiller/cubems-smart-building-energy-and-iaq-data

[5] https://www.kaggle.com/claytonmiller/cubems-smart-building-energy-and-iaq-data

Algorithm 1 MORSTS

Require: For each batch $\mathcal{D}^t, t = 1, 2, ...$, its instances (training data) are tuples $\left(x^t(i); y^t(i)\right)$, where $x^t(i) \in \mathbb{R}^{g'}, y^t(i) \in \mathbb{R}^g, i = 1, ..., m^t$.
Require: Parameters a (slope) and b (inflection point) of the sigmoid function
Require: T: Maximum number of submodels
Require: k: Parameter of the k-fold validation
Require: V, Parameter of submodel validation
Require: $ESM \leftarrow \varnothing$, Set of submodels
Ensure: H^t for t = 1,2,3,....

for t = 1,2,3,.... **do**
  if t = 1 **then**
    $algo \leftarrow$ randomly select an algorithm $\in EAR$,
    $h^1 \leftarrow$ train a submodel according to $algo$ with $\mathcal{D}^1$
    $ESM \leftarrow ESM \cup h^1$
    $H^1 \leftarrow h^1$
  else
    $\left[\hat{y}_1^t(i), .., \hat{y}_g^t(i)\right] \leftarrow H^{t-1}(x^t(i))$, for $i = 1, ..., m^t$
    $err^t(i) \leftarrow \frac{1}{g}\sum_{c=1}^{g} |\hat{y}_c^t(i) - y_c^t(i)|$, for $i = 1, ..., m^t$
    $\Theta^t(i) \leftarrow err^t(i) / \sum_{i=1}^{m^t} err^t(i)$ for $i = 1, ..., m^t$
    for $\forall h^j \in ESM$ **do**
      $\left[\hat{y}_1^t(i), .., \hat{y}_g^t(i)\right] \leftarrow h^k(x^t(i))$, for $i = 1, ..., m^t$
      $err^t(i) \leftarrow \frac{1}{g}\sum_{c=1}^{g} |\hat{y}_c^t(i) - y_c^t(i)|$, for $i = 1, ..., m^t$
      $\varepsilon_{h^j}^t \leftarrow \frac{1}{m^t}\sum_{i=1}^{m^t} \Theta^t(i) \cdot err^t(i)$
    end for
    $\varepsilon_{H^{t-1}}^t \leftarrow \frac{1}{m^t}\sum_{i=1}^{m^t} \Theta^t(i) \cdot err^t(i)$
    $r = 1$
    while True **do**
      $algo \leftarrow$ randomly choose an algorithm $\in EAR$.
      $\left[\hat{y}_1^t(i), .., \hat{y}_g^t(i)\right]$ for $i = 1, ..., m^t \leftarrow$ k-fold-cross-validation $(k, algo, \mathcal{D}^t)$
      $err^t(i) \leftarrow \frac{1}{g}\sum_{c=1}^{g} |\hat{y}_c^t(i) - y_c^t(i)|$, for $i = 1, ..., m^t$
      $\varepsilon_{h^t}^t \leftarrow \frac{1}{m^t}\sum_{i=1}^{m^t} \Theta^t(i) \cdot err^t(i)$
      if $\varepsilon_{h^t}^t < \varepsilon_{H^{t-1}}^t \cdot V * *(r-1)$ **then**
        $h^t \leftarrow$ train a submodel according to $algo$ with $\mathcal{D}^t$
        $ESM \leftarrow ESM \cup h^t$
        break
      else
        $r \leftarrow r + 1$
      end if
    end while
    if $|ESM| > T$ **then**
      $ESM \leftarrow ESM - h_{j'}$, where $\varepsilon_{h^{j'}}^t = \arg\max_{h^j \in ESM} \varepsilon_{h^j}^t$
    end if
    for $\forall h^j \in ESM$ **do**
      $W_{h^j}^t \leftarrow \log_e\left(\frac{1}{\varepsilon_{h^j}^t \cdot sigm(a,b,t-j)}\right)$
    end for
    $H^t \leftarrow \sum_{h^j \in ESM}(W_{h^j}^t \cdot h^j) / \sum_{h^j \in ESM} W_{h^j}^t$
  end if
end for

to December 2019. The sampling interval is 1 minute (table 3). Fig. 3-(b) and Fig. 4-(b) show respectively the imbalanced and non-stationarity of CU-BEMS dataset.

Table 4 shows that there is an autocorrelation of the energy consumption data of the same appliances, for the UK-DALE data. Autocorrelation is due to the fact that the current state of a device can be correlated with the previous state, i.e., there is a **time dependence**. As the lag (i.e., time difference) increases, the autocorrelation decreases.

FIGURE 3. Normalized data from 4 devices over a one month from (a) UK-DALE and (b) CU-BEMS. The green triangle represents the median and the yellow vertical line represents the mean.

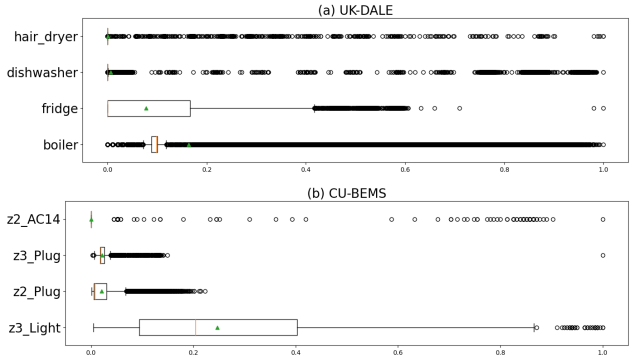


FIGURE 4. Imbalanced data from 2 devices during 2 days.

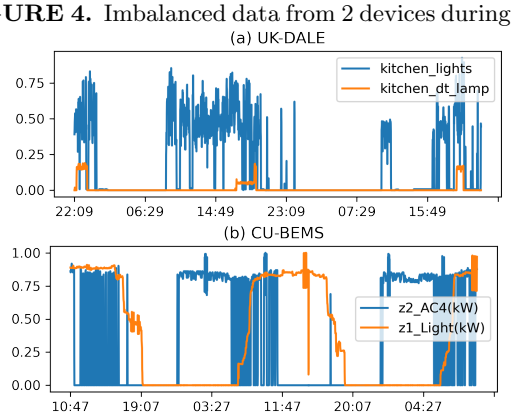


TABLE 2. Sample of normalised UK-DALE data (part of the appliances, 2012-Nov)

Time	hoover	coffee_machine	kitchen_radio	kitchen_dt_lamp	bedroom_d_lamp	gigE_&_USBhub	samsung_charger	adsl_router	bedroom_ds_lamp	lighting_circuit	kitchen_lamp2	livingroom_s_lamp	iPad_charger	utilityrm_lamp	kitchen_phone&stereo
Nov.22 22:09:06	0.0	0.0	0.01	0.00	0	0	0	0.70	0	0.68	0.71	0	0	0	0.51
Nov.22 22:10:06	0.0	0.0	0.01	0.00	0	0	0	0.78	0	0.74	0.71	0	0	0	0.56
Nov.22 22:11:06	0.0	0.0	0.01	0.00	0	0	0	0.70	0	0.76	0.80	0	0	0	0.49
Nov.22 22:12:06	0.0	0.0	0.01	0.07	0	0	0	0.78	0	0.70	0.71	0	0	0	0.55
Nov.22 22:13:06	0.0	0.0	0.01	0.00	0	0	0	0.70	0	0.72	0.71	0	0	0	0.48
Nov.22 22:14:06	0.0	0.0	0.00	0.00	0	0	0	0.70	0	0.63	0.70	0	0	0	0.48
Nov.22 22:15:06	0.0	0.0	0.00	0.00	0	0	0	0.70	0	0.51	0.80	0	0	0	0.48
Nov.22 22:16:06	0.0	0.0	0.00	0.00	0	0	0	0.78	0	0.59	0.70	0	0	0	0.54
Nov.22 22:17:06	0.0	0.0	0.00	0.00	0	0	0	0.78	0	0.54	0.78	0	0	0	0.54
Nov.22 22:18:06	0.0	0.0	0.01	0.00	0	0	0	0.78	0	0.40	0.80	0	0	0	0.68

TABLE 3. Sample of normalised CU-BEMS data (part of the appliances, 2019-JAN)

Time	Z1_AC1	Z1_Light	Z1_Plug	Z2_AC1	Z2_AC5	Z2_AC6	Z2_AC8
JAN.01 22:15	0.0005	0.0004	0.0153	0.0211	0.0819	0.1394	0.1424
JAN.01 22:16	0.0004	0.0004	0.0153	0.0213	0.0819	0.1374	0.1408
JAN.01 22:17	0.0004	0.0004	0.0153	0.0213	0.0819	0.1130	0.1161
JAN.01 22:18	0.0005	0.0004	0.0153	0.0213	0.0799	0.1360	0.1393
JAN.01 22:19	0.0004	0.0004	0.0153	0.0213	0.0799	0.1371	0.1409
JAN.01 22:20	0.0004	0.0004	0.0153	0.0213	0.0799	0.1371	0.1407
JAN.01 22:21	0.0005	0.0004	0.0153	0.0211	0.0800	0.1382	0.1407
JAN.01 22:22	0.0004	0.0004	0.0153	0.0211	0.0800	0.1382	0.1424
JAN.01 22:23	0.0005	0.0004	0.0153	0.0213	0.0719	0.1382	0.1424
JAN.01 22:24	0.0005	0.0004	0.0153	0.0213	0.0809	0.1394	0.1439

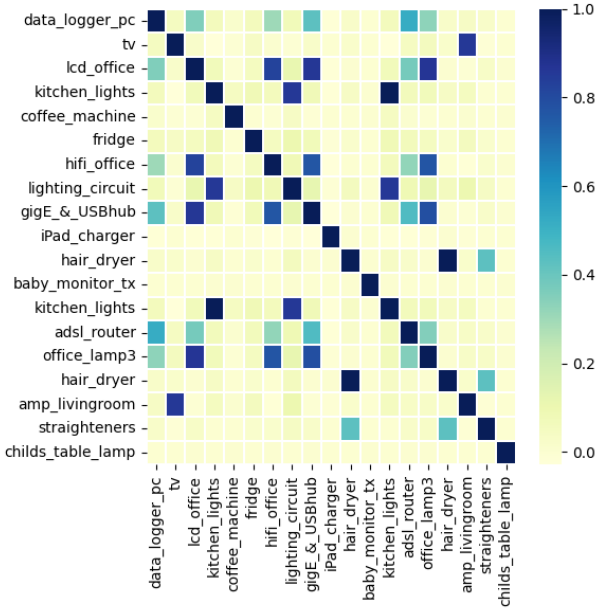
TABLE 4. Autocorrelation for some UK-DALE devices (part of the appliances), with sampling occurring from November to December 2013, lags from 0 to 10.

lag	hoover	coffee_machine	kitchen_radio	kitchen_dt_lamp	bedroom_d_lamp	gigE_&_USBhub	samsung_charger	adsl_router	bedroom_ds_lamp	lighting_circuit	kitchen_lamp2	livingroom_s_lamp	iPad_charger	utilityrm_lamp	kitchen_phone&stereo
0	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1	0.87	0.78	0.22	0.97	0.96	0.99	0.75	0.37	0.97	0.96	0.98	0.99	0.39	0.88	0.99
2	0.76	0.53	0.22	0.94	0.94	0.99	0.75	0.44	0.94	0.92	0.98	0.99	0.39	0.77	0.99
3	0.70	0.34	0.21	0.91	0.92	0.98	0.74	0.47	0.91	0.90	0.98	0.98	0.38	0.71	0.99
4	0.66	0.17	0.20	0.89	0.89	0.98	0.73	0.42	0.87	0.87	0.98	0.98	0.38	0.69	0.99
5	0.62	0.12	0.20	0.86	0.87	0.97	0.73	0.41	0.85	0.85	0.98	0.97	0.38	0.67	0.99
6	0.58	0.12	0.20	0.83	0.85	0.97	0.72	0.42	0.82	0.83	0.98	0.96	0.38	0.65	0.99
7	0.56	0.09	0.19	0.81	0.83	0.96	0.71	0.40	0.79	0.81	0.98	0.96	0.38	0.63	0.99
8	0.54	0.08	0.19	0.78	0.80	0.96	0.71	0.38	0.76	0.79	0.97	0.95	0.37	0.61	0.99
9	0.51	0.06	0.19	0.75	0.78	0.96	0.70	0.39	0.74	0.78	0.97	0.95	0.37	0.59	0.99
10	0.50	0.06	0.18	0.72	0.76	0.95	0.69	0.38	0.72	0.76	0.97	0.94	0.37	0.58	0.99

For the UK-DALE dataset, we configured MORSTS to take as input the energy consumption data of the 5 minutes before t for all devices (i.e., a sliding window with a width of 5 in Fig. 1). Table 4 shows that: 1) when the lag is less than 5, the autocorrelation of some devices decreases; 2) when the lag is greater than 5, the autocorrelation coefficients of almost all devices remain stable. Considering that the number of devices is 48, the length of the input vector g' is 240. Similarly, for CU-BEMS dataset, MORSTS has been configured to take as input the energy consumption data of the 3 minutes preceding t for all devices (i.e., a sliding window with a width of 3 in Fig. 1). Considering that the number of devices is 22, the length of the input vector g' is 66.

Fig. 5 shows the correlation between the energy consumption data of UK-DALE devices. For example, there is a significant positive correlation between the hifi_office and lcd_office. It is likely that hifi_office and lcd_office are used during the same period.

FIGURE 5. Correlation between some appliances, UK-DALE.



5.2. Pre-processing of data

Two main pre-processing have been applied before conducting our experiments:

- **Normalization**: When having multiple output functions, normalization is essential, as the loss function is calculated for several variables [1]. The aim of normalization is to bring all values of the variable back between 0 and 1, while maintaining a certain distance between these values. So, for both datasets, the power consumption values have been normalised for each device using equation 2 where: x is an original value of a given device, $\max(x)$ and $\min(x)$ are the max value and min value from the same device, and x' is the corresponding normalized value.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{2}$$

- **Elimination of collinearity**: Correlation between devices can lead to collinearity between the explanatory variables (inputs), which can affect the accuracy of the prediction model [25]. According to [25], there are three strategies for solving the collinearity problem: 1) eliminating collinearity before modeling, 2) modeling with latent variables, and 3) modeling insensitive to collinearity. Removing collineraity before model building is the most widely used method, and experiments described in [25] have shown that the latter two types of method have no relative advantage in terms of accuracy. Principal component analysis (PCA) is the most commonly used method for eliminating collinear-

ity prior to modeling. The main idea behind PCA is to go from N dimensions (variables) to N' decorrelated dimensions (variables) [26], N' smaller than N. PCA not only reduces training time and model complexity, but also enables the model to achieve better prediction performance [26]. That's why, we applied Principal Component Analysis (PCA) for both datasets. For UK-DALE dataset, the length of the input vector g' has been reduced from 240 to 25 after PCA . Fig. 6 shows that: 1) when the number of generated dimensions of the inputs exceeds 25, the share of variance explained by the added dimension tends to zero; 2) the share of cumulative explanatory variance of the 25 dimensions is 94%, which exceeds 85% and is therefore expected to explain almost all of the variance [27]. Similarly, for CU-BEMS, the length of the input vector has been reduced from 66 to 10 after PCA (see Fig. 7).

5.3. Evaluation metrics

For evaluation purposes, we used *Accuracy* and *Efficiency* metrics:

- Prediction **Accuracy** is the average root mean squared error (aRMSE) of all variables measures the global accuracy (equation 3).
- Model **Efficiency** is the average running time for each 10,000 data (vector).
  Numbers discussed below have been obtained in using a MAC mini 2014, Core i5 chip, 8GB RAM, with Python 3.7.

$$\text{aRMSE} = \frac{1}{g}\sum_{f=1}^{g} \text{RMSE for } f\text{th variable} \qquad (3)$$

5.4. Super-parameters

Several super-parameters have been used:

Type of submodels The submodels have been randomly selected from multi-output models in libraries (such as Sklearn [6]): Ridge regression, Decision Trees, Random Forest and LSTM.

a and b are the parameters of Sigmoid function, which assigns time weights to the submodels (the closer they are, the greater it is). For UK-DALE dataset, $a = 0.35$ (slope) (optimal value between 0.05 and 1 by steps of 0.1 for a best aRMSE) and $b = 1$ (optimal value between 1 and 5 by step of 1 for a best aRMSE). For CU-BEMS dataset, we set $a = 0.4$ and $b = 2$.

$|\mathcal{D}^t|$ *(Batch size)* Batch size is essential for accurate models, but difficult to determine [28]. By varying the batch size as illustrated in Fig. 8-(a), the optimal value of the batch size for UK-DALE dataset is 10080 and it is equal to 7200 for CU-BEMS (Fig. 9-(a)).

K-fold cross-validation (KFCV) MORSTS uses KFCVto evaluate the weight of the emerging submodel, to mitigate the risk of over-learning of submodels and their negative effects. According to [29], when the amount of available data is large, the value of k can be relatively small to save time costs. However, it is not specified in [29] which criteria to be used. For UK-DALE, we set $k = 3$ because the accuracy is stable and the execution time increases steadily with k (as shown in Fig. 8-(b)). Similarly, for the CU-BEMS dataset, we set $k = 4$ (Fig. 9-(b)).

Submodel validation parameter (V) In MORSTS, a new approved submodel h^t must be generated under the condition $\varepsilon^t_{h^t} < \varepsilon^t_{H^{t-1}} \cdot V * *(r-1)$. The larger is V value , the higher is the overfitting risk. The smaller is V value, the bigger is the number of attempts (i.e., computational cost) will be made to obtain a new validated submodel. We set $V = 1.2$ for UK-DALE, which is the most optimal value between 1.1 and 1.9, as illustrated in Fig. 8-(c), and $V = 1.4$ for CU-BEMS dataset (Fig. 9-(c)).

FIGURE 6. Variance of inputs (UK-DALE data) by PCA.

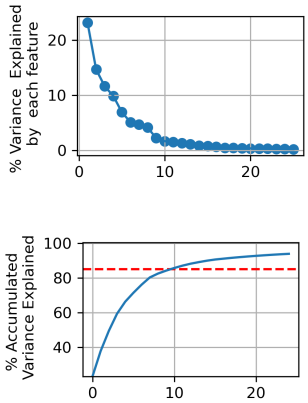
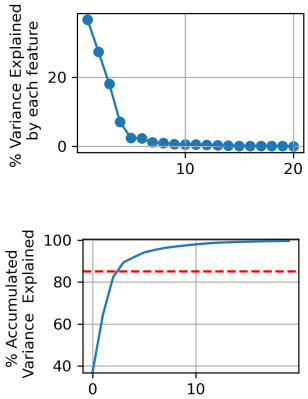


FIGURE 7. Variance of inputs (CU-BEMS data) by PCA.



[6] https://scikit-learn.org/stable/

Maximum number of submodels (T) The maximum number of submodels for both datasets is $T = 12$ which is the optimal value to get the best accuracy (Fig. 8-(d) and Fig. 9-(d)).

5.5. MORSTS Assessment

We compared MORSTS with MORSTS$_{noKFCV}$, its version without KFCV, to check whether cross-validation effectively controls overfitting or not. We also compared MORSTS with some multiple output models such as Decision Trees, Random Forest, Ridge regression, and LSTM. For UK-DALE dataset (resp. CU-BEMS), the latter models use the first 10080 (resp. 7200) instances for initialization, and the remaining data for testing, with no model update and no cross validation.

5.6. Results

Overfitting Figs. 10-(a) and (b) (resp. Figs. 11-(a) and (b)) show the distance between the Training Error and the Testing Error which is the measure of overfitting in UK-DALE (resp. CU-BEMS), corresponding to the blue areas: the larger is the blue area, the greater is overfitting. It is worth noticing that, considering the flow of data, for a newcomer submodel on the t_{th} batch, its learning error is on the tth batch, and its generalization error is on the $t+1$th batch. We observe that the number of MORSTS blue areas is significantly smaller than MORSTS$_{noKFCV}$ one. This is due to the fact that MORSTS has a process based on k-fold cross-validation, which limits the risk of overfitting. Fig. 10-(b) also shows that the MORSTS submodels trained on the 4th, 14th, 35th, 56th, 65th batches are relatively large over-fitted. The most likely explanation is that the house was unoccupied during this time and the appliances were switched off (almost no energy consumption). For the same reasons, MORSTS has very low aRMSE values for these batches (4th, 14th, 35th, 56th, 65th) as shows Fig. 10-(c). Although the MORSTS submodels born in the 4th, 14th, 35th, 56th and 65th batches suffered from overfitting, in the UK-DALE dataset, the MORSTS's aRMSE values however did not receive a significant negative impact for the successor batches (i.e., 5th, 15th, 36th, 57th and 66th) as shown in Figure 10-(c). Since these submodels with the risk of overfitting are given very low weights, this does not affect the voting results, thanks to the k-fold cross-validation. Fig. 10-(a) shows that MORSTS$_{noKFCV}$ has a significant overfitting in 3rd, 7th, and 10th batches, in the UK-DALE dataset. As a result, in the following batches, the prediction accuracy of MORSTS$_{noKFCV}$ decreases significantly (i.e., the blue curve increases explicitly in Fig. 10-(c)). This is due to the fact that the submodels trained on the 3rd, 7th, and 10th batches were incorrectly assigned high weights based on their learning errors. In the UK-DALE dataset, we also observed that the overfitting of the MORSTS$_{noKFCV}$ newborn submodel persisted from batch 15 onward, but there was a trend toward improved prediction accuracy for MORSTS$_{noKFCV}$ (i.e., the blue curve has a downward trend and then remains stable in Fig. 10-(a)). One possible explanation is that MORSTS$_{noKFCV}$ has collected a sufficient number of submodels with appropriate weights since the 15th batch, so the negative effects of overfitting newcomer submodels are controlled. To summarize, our experiments on CU-BEMS and UK-DALE datasets show that MORSTS performs well with respect to prediction accuracy, thanks to the control of overlearning. It is also important to note that CU-BEMS data come from office buildings, not houses.

Accuracy We evaluated the accuracy of MORSTS and compared it with other baseline models. We note that MORSTS has a better accuracy in most of the batchs for both datasets as illustrated in Fig. 10 and Fig. 11. The average accuracy for all the the models is described in table 5 showing that MORSTS outperforms the other models thanks to cost-sensitive and KFCV strategies. We also compared MORSTS with MORSTS$_{noKFCV}$, its version without KFCV. The outcome was that multiple validation helps improving global accuracy. Finally, we note that ensemble models combined with cost-sensitive strategy perform better than baseline models (Decision Tree, Random forest, Ridge Regression, LSTM) in case of imbalanced and non-stationary STS.

Single output vs. Multi-output We compared the accuracy (aRMSE) of the original multi-output MORSTS and with its single-output version, i.e., all MORSTS submodels are used with their single output versions. Regarding the datasets, we used a sample of highly correlated sensors and and weakly correlated ones. As illustrated in Fig. 5, we selected 5 pairs of highly correlated UK-DALE sensors: 1) multiple outputs for each pair of sensors and 2) separate single outputs for individual sensors. We used the same inputs for both predictions i.e.,

past energy consumption data obtained with a 5 minutes sliding window for 48 appliances with at 1 minute delay, reduced to 25 dimensions after PCA. Table 6 shows that, in presence of correlation, original MORSTS is in most cases more accurate than single-output models. Similarly, we selected 5 pairs of weakly correlated sensors (Fig. 5). Table 7 shows that, for some pairs of weakly correlated sensors, the multi-output model does not obtain better accuracy, compared to the single-output model. This situation supports the statement of [1] regarding the fact that correlated output may improve prediction accuracy.

Execution time We compared the different models in terms of time execution for 10 thousand predictions as illustrated in Table 8. For UK-DALE data, we note that MORSTS is time-consuming (time = 2s) in

TABLE 5. Accuracy of Baseline Models.

	MORSTS	MORSTS noKFCV	Decision Tree	Random forest	Ridge Regression	LSTM
aMRSE, UK-DALE	0.019	0.024	0.034	0.024	0.034	0.057
aMRSE, CU-BEMS	0.057	0.066	0.134	0.107	0.159	0.172

FIGURE 8. Super-parameters of MORSTS with UK-DALE.

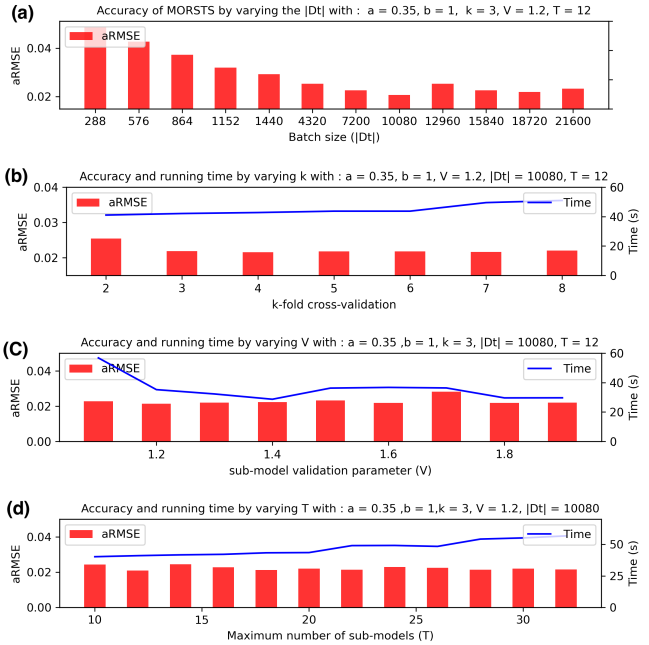


FIGURE 9. Super-parameters of MORSTS with CU-BEMS.

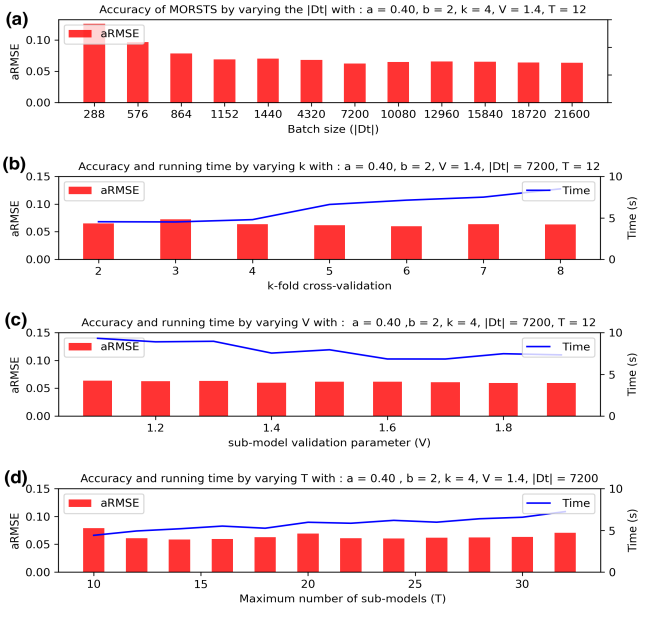


FIGURE 10. Overfitting and aRMSE of MORSTS with UK-DALE.

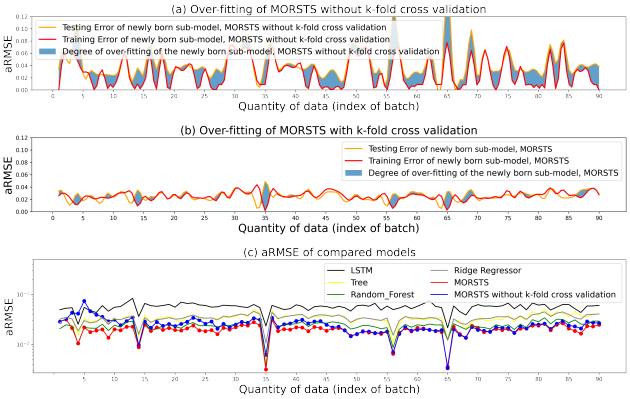
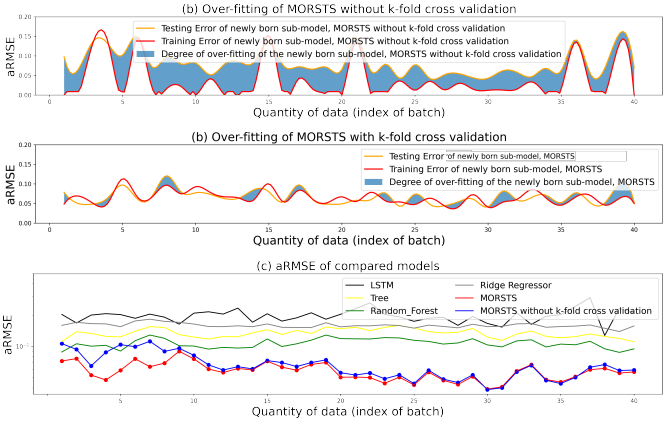


FIGURE 11. Overfitting and aRMSE of MORSTS with CU-BEMS.



comparison to Decision Trees (0.28s), Random Forests (0.27s), Ridge Regression (0.2s) and LSTM (0.6s). This is because each MORSTS prediction requires all submodels' predictions followed by voting. We observe that 10,000 predictions take 2 seconds which means that each prediction takes an average of 0.0002 seconds, which is far below the sampling time of the dataset used in this experiment (1 minute). Hence, we may conclude that MORSTS can provide real-time predictions.

In terms of accuracy, Table 5 shows that Random Forest (RF) has the best accuracy with minimal computing time for both datasets (see Table 8). Although Random forest is behind MORSTS in terms of accuracy, RF may perform better assuming it could make use of online updating and cost-sensitive

TABLE 6. Accuracy of Multi vs. Single-output: case of highly correlated sensors

	Multi-output adaptation	Transformation into several single output models	Relative advantage
#43 data_logger_pc #18 adsl_router	0.0025	0.0026	+0.0001
#50 office_lamp3 #21 gigE_&_USBhub	0.0067	0.0073	+0.006
#8 kitchen_lights #25 lighting_circuit	0.0594	0.0652	+0.0058
#39 hair_dryer #40 straighteners	0.0089	0.0089	0
#17 amp_livingroom # 7 tv	0.039	0.041	+0.002

TABLE 7. Accuracy of Multi vs. Single-output: case of weakly correlated sensors

	Multi-output adaptation	Transformation into several single output models	Relative advantages
#27 iPad_charger #52 baby_monitor_tx	0.0203	0.0195	-0.0008
#39 hair_dryer #12 fridge	0.0156	0.0156	0
#44 childs_table_lamp #18 adsl_router	0.0035	0.0028	-0.007
#6 dishwasher #36 coffee_machine	0.0176	0.0182	+0.006
#15 hifi_office #8 kitchen_lights	0.0380	0.0408	+0.028

strategies,

in presence of imbalanced and non-stationary data stream.

In addition, Random forest can replace MORSTS when 1) the sampling rate is very small and 2) the requirement for prediction accuracy is not very demanding, to avoid the delay in prediction due to the update time of MORSTS. Conversely, MORSTS can provide optimal accuracy when the sampling rate is significantly larger than the time required for the update of MORSTS (e.g. in this experiment).

We evaluated the update run times of MORSTS and its version without KFCV MORSTS$_{noKFCV}$. We noted that MORSTS (38s) is more expensive in terms of updating time than MORSTS$_{noKFCV}$ (14s) because it is cross-validation based. In the previous experiments, we have found that 1) KFCV significantly improves the performance of MORSTS in the early stages of the online process , and 2) KFCV becomes less useful after MORSTS has accumulated enough submodels. Therefore, a potential optimization scheme is that if the prediction accuracy of MORSTS tends to be stable, then newcomer submodels are exempted from k-fold cross-validation. The advantage of this strategy is to reduce the use of k-fold cross-validation, which means that the computation time of MORSTS would be reduced without affecting the accuracy of the predictions.

TABLE 8. Running time for 10K predictions. UK-DALE and CU-BEMS.

	MORSTS	MORSTS noCVKF	Decision Tree	Random forest	Ridge Regression	LSTM
for prediction (s), UK-DALE	2	2	0.28	0.27	0.2	0.6
for updating (s), UK-DALE	38	14	-	-	-	-
for prediction (s), CU-BEMS	0.51	0.49	0.07	0.08	0.07	0.22
for updating (s), CU-BEMS	7	3.5	-	-	-	-

Our experiments showed that MORSTS performs better than baseline models on imbalanced data stream and handles overfitting but remains expensive due to KFCV and models' update.

6. CONCLUSION AND FUTURE WORK

In this paper, we described MORSTS, an online ensemble regression model for making predictions on non-stationary and imbalanced data streams. MORSTS adopts a cost-sensitive strategy for imbalanced data, and performs k-fold cross-validation (KFCV) to handle submodels' overfitting.

Experiments conducted with two real datasets from smart buildings show that (1) MORSTS improves prediction accuracy when there exist a correlation between output variables compared to some renown and multi-output models; (2) KFCV significantly improves the accuracy at the early stages of the online process. However, MORSTS execution time is costly, due to voting and KFCV procedures.

Lessons learnt from MORSTS already emphasized the impact of KFCV and multiple output correlations. For future work, we target further improvement of MORSTS by combining different state of the art approaches. Among other approaches that are worth to consider one can cite: rule-based algorithms [22], learning automatically correlations between multiple outputs [10] and under/over sampling algorithms [18]. Indeed, the correlation between output variables can vary over time. In this case, the dynamic transformation of multiple-output problems into several multiple-output sub-problems [22, 10] represents a research direction we wish to explore. Thus, one (or more) submodel(s) is (are) formed to correspond to each multiple-output sub-problem. However, this requires 1) automatically and dynamically (without human intervention) planning multiple-output sub-problems in a data stream; for example, it is better to avoid some variables being covered by only a few sub-problems and others by too many; 2) training the corresponding prediction model if a new multiple-output sub-problem appears and avoiding repeated access to historical data; 3) knowing how these submodels, with different weights and output structures, will vote to produce the final result. In addition, when the data are imbalanced, how to apply a cost-sensitive strategy to submodels with

different output structures is a line of research to be explored. Finally, we aim to consider extra information related to the sensors and to the energy consumption such as weather (e.g. temperature, barometric pressure, etc.) in order to analyze their impact on the regression outputs.

REFERENCES

[1] Borchani, H., Varando, G., Bielza, C. & Larrañaga, P.(2015) A Survey on Multi-Output Regression. *Wiley Int. Rev. Data Min. And Knowl. Disc.*. **5**, 216-233 , https://doi.org/10.1002/widm.1157

[2] Ribeiro, R. & Moniz, N. (2020) Imbalanced regression and extreme value prediction. *Mach. Learn.*. **109**, 1803-1835 , https://doi.org/10.1007/s10994-020-05900-9

[3] Wang, Y., Chen, Q., Kang, C., Xia, Q. & Luo, M. (2016) Sparse and redundant representation-based smart meter data compression and pattern extraction. *IEEE Transactions On Power Systems*. **32**, 2142-2151

[4] Ji, Y., Buechler, E. & Rajagopal, R. (2019) Data-driven load modeling and forecasting of residential appliances. *IEEE Transactions On Smart Grid*. **11**, 2652-2661

[5] Krawczyk, B. (2016) Learning from imbalanced data: open challenges and future directions. *Progress In Artificial Intelligence*. **5**, 221-232

[6] Chawla, N., Bowyer, K., Hall, L. & Kegelmeyer, W. (2002) SMOTE: synthetic minority over-sampling technique. *Journal Of Artificial Intelligence Research*. **16** pp. 321-357

[7] Mani, I. & Zhang, I. (2003) kNN approach to unbalanced data distributions: a case study involving information extraction. *Proceedings Of Workshop On Learning From Imbalanced Datasets*. **126**

[8] Gomes, H., Montiel, J., Mastelini, S., Pfahringer, B. & Bifet, A. (2020) On ensemble techniques for data stream regression. *2020 International Joint Conference On Neural Networks (IJCNN)*. pp. 1-8

[9] Gomes, H., Read, J., Bifet, A., Barddal, J. & Gama, J.(2019) Machine learning for streaming data: state of the art, challenges, and opportunities. *ACM SIGKDD Explorations Newsletter*. **21**, 6-22

[10] Yu, H., Lu, J. & Zhang, G. (2022) MORStreaming: A Multioutput Regression System for Streaming Data. *IEEE Trans. Syst. Man Cybern. Syst.*. **52**, 4862-4874

[11] Mitchell, T. (1997)& Others Machine learning. (McGraw-hill New York,1997)

[12] Zuo, J., Zeitouni, K. & Taher, Y. (2019) Time series meet data streams: Perspectives of the interdisciplinary collision and applications. *PhD Symposium, Actes De La Conférence BDA*.

[13] Soares, S. & Araújo, R. (2015) An on-line weighted ensemble of regressor models to handle concept drifts. *Engineering Applications Of Artificial Intelligence*. **37** pp. 392-406

[14] Kadlec, P. & Gabrys, B. (2011) Local learning-based adaptive soft sensor for catalyst activation prediction. *AIChE Journal*. **57**, 1288-1301

[15] Kolter, J. & Maloof, M. (2005) Using additive expert ensembles to cope with concept drift. *Proceedings Of The 22nd International Conference On Machine Learning*. pp. 449-456

[16] Xiao, J., Xiao, Z., Wang, D., Bai, J., Havyarimana, V. & Zeng, F. (2019) Short-term traffic volume prediction by ensemble learning in concept drifting environments. *Knowl. Based Syst.*. **164** pp. 213-225 , https://doi.org/10.1016/j.knosys.2018.10.037

[17] Elwell, R. & Polikar, R. (2009) Incremental learning of variable rate concept drift. *International Workshop On Multiple Classifier Systems*. pp. 142-151

[18] Aminian, E., Ribeiro, R. & Gama, J. (2021) Chebyshev approaches for imbalanced data streams regression models. *Data Min. Knowl. Discov.*. **35**, 2389-2466

[19] Ikonomovska, E., Gama, J. & Dzeroski, S. (2011) Incremental multi-target model trees for data streams. *Proceedings Of The 2011 ACM Symposium On Applied Computing*. pp. 988-993

[20] McWilliams, B. & Montana, G. (2010) Sparse partial least squares regression for on-line variable selection with multivariate data streams. *Statistical Analysis And Data Mining: The ASA Data Science Journal*. **3**, 170-193

[21] Li, C., Wei, F., Dong, W., Wang, X., Liu, Q. & Zhang, X. (2018) Dynamic structure embedded online multiple-output regression for streaming data. *IEEE Transactions On Pattern Analysis And Machine Intelligence*. **41**, 323-336

[22] Duarte, J. & Gama, J. (2015) Multi-target regression from high-speed data streams with adaptive model rules. *2015 IEEE International Conference On Data Science And Advanced Analytics (DSAA)*. pp. 1-10

[23] Aguiar, G., Krawczyk, B. & Cano, A. A survey on learning from imbalanced data streams: taxonomy, challenges, empirical study, and reproducible experimental framework. (2022) *CoRR*. **abs/2204.03719**

[24] Yang, Y., Zha, K., Chen, Y., Wang, H. & Katabi, D. (2021) Delving into Deep Imbalanced Regression. *Proceedings Of The 38th International Conference On Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. **139** pp. 11842-11851

[25] Dormann, C., Elith, J., Bacher, S., Buchmann, C., Carl, G., Carre, G., Garcia Marquez, J., Gruber, B., Lafourcade, B., Leitao, P., Münkemüller, T., Mcclean, C., Osborne, P., Reineking, B., Schroeder, B., Skidmore, A., Zurell, D. & Lautenbach, S. (2013) Collinearity: a review of methods to deal with it and a simulation study evaluating their performance. *Ecography*. **36**, 27 - 46 , https://hal.inrae.fr/hal-02651746

[26] Xu, X., Liang, T., Zhu, J., Zheng, D. & Sun, T. (2019) Review of classical dimensionality reduction and sample selection methods for large-scale data processing. *Neurocomputing*. **328** pp. 5-15

[27] Xue-Hua, Z., Xu-Juan, M., Zhen-Gang, Z. & Zheng, H. (2019) Research on prediction method of reasonable cost level of transmission line project based on PCA-LSSVM-KDE. *Mathematical Problems In Engineering*.

[28] Gomes, H., Barddal, J., Enembreck, F. & Bifet, A. (2017) A survey on ensemble learning for data stream classification. *ACM Computing Surveys (CSUR)*. **50**, 1-36

[29] Hawkins, D. (2004) The problem of overfitting. *Journal Of Chemical Information And Computer Sciences*. **44**, 1-12