Received: 21 March 2022 | Accepted: 24 July 2022

DOI: 10.1002/int.22982

RESEARCH ARTICLE

WILEY

Piecewise forecasting of nonlinear time series with model tree dynamic Bayesian networks

David Quesada[1] | **Concha Bielza**[1] | **Pedro Fontán**[2] | **Pedro Larrañaga**[1]

[1]Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, Madrid, Spain

[2]REPSOL Technology Lab, Madrid, Spain

Correspondence
David Quesada, Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, Madrid, Spain.
Email: dquesada@fi.upm.es

Funding information

Ministerio de Ciencia, Innovación y Universidades, Grant/Award Number: PID2019-109247GB-I00; Fundación BBVA, Grant/Award Number: Score-based nonstationary temporal Bayesian networks. Applications in climate and neuroscience (BAYES-CLIMA-NEURO)

Abstract

When modelling multivariate continuous time series, a common issue is to find that the original processes that generated the data are nonlinear or that they drift away from the original distribution as the system evolves over time. In these scenarios, using a linear model such as a Gaussian dynamic Bayesian network (DBN) can result in severe forecasting inaccuracies due to the structure and parameters of the model remaining constant without considering either the elapsed time or the state of the system. To approach this problem, we propose a hybrid model that combines a model tree with DBNs. The model first divides the original data set into different scenarios based on the splits made by the tree over the system variables and then performs a piecewise regression in each branch of the tree to obtain nonlinear forecasts. The experimental results on three different datasets show that our model outperforms standard DBN models when dealing with nonlinear processes and is competitive with state-of-the-art time series forecasting methods.

KEYWORDS

dynamic Bayesian networks, forecasting, piecewise regression, regression trees, time series

1 | INTRODUCTION

One of the most common types of data faced when modelling industrial processes are multivariate time series (TS).[1] With the increase in the number of sensors and their widespread usage in monitoring industrial processes, understanding the interactions between variables in a system and treating the TS resulting from these sensors has become a pressing matter.

In a multivariate setting, we must take into account the effects that fluctuations in a variable have on the rest of the system. This means that we face a set of coupled univariate TS where variations in one series can generate linear or nonlinear changes in the others. As a result, modelling a multivariate process over time requires updating the state of the system and forecasting the evolution of all variables simultaneously.

One kind of model that can deal with this type of data is dynamic Bayesian networks (DBNs).[2] A DBN models the probabilistic conditional independence relationships of the variables in data over time, representing the effect that past instants have on the present. This model also allows for a better understanding of a system through its graphical representation and can perform probabilistic reasoning over any desired set of objective variables given that others have been observed. In addition, a DBN model can work both as a forecasting tool by predicting the evolution of all variables in a system and as a generative model by simulating the behaviour of a process given some fixed evidence.

In recent years, DBN models have seen a lot of use in industrial settings due to their characteristics as TS forecasting models and their interpretability, which has changed DBNs into more general use models. They have been applied to stock market forecasting,[3] to ecosystem changes prediction based on climate variations,[4] to topic-sentiment evolution analysis over time,[5] to assess the remaining useful life of structures,[6,7] to monitor aircraft wing cracks evolution over time[8] and to identify abnormal events during cyber security threats,[9] among others. However, in a continuous case, where Gaussianity is typically assumed, DBNs present some drawbacks: DBN models are inherently linear models, and they do not allow the insertion of discrete variables without the introduction of additional constraints.[10] Over time, industrial processes commonly follow complex nonlinear relationships or have changing distributions of the variables as the system evolves, incurring in a phenomenon called *concept drift*.[11] In both scenarios, a traditional DBN presents weaknesses inherent to the model that will result in severe inaccuracies when modelling and forecasting this kind of pattern. In the aforementioned applications, Gaussian DBNs are used to fit nonlinear problems. In those cases, the authors have to either assume Gaussianity in their experiments, discretize the variables to be able to apply discrete DBNs or modify the DBN architecture with nonlinear conditional probability distributions that do not allow exact inference and need Markov Chain Monte Carlo sampling.[5]

To address the linearity issue of Gaussian DBNs, we propose a new hybrid model called model tree dynamic Bayesian network (mtDBN). This model combines a classification and regression tree (CART)[12] with DBNs in a manner similar to that of model trees.[13] First, a tree is fitted over the data with the desired variables, with the possibility of adding the elapsed time as a variable too. This tree structure is used to classify all instances of the data set into different subpopulations depending on which leaf node they correspond to. Finally, a DBN model is fitted to each of the subsets of data in the leaf nodes. This way, we obtain several DBN models tuned specifically for certain contexts of the feature space defined by the tree splits instead of a unique global network. When performing forecasting, we first classify the current state of the system with the tree structure, and then we perform inference with the appropriate model. This

results in a piecewise regression,[14] which is closer than a linear model to the real behaviour of a nonlinear or nonstationary system. Another advantage of this model is that in the presence of discrete variables, one can use all or several of them in the construction of the tree structure and then train Gaussian DBNs on each leaf node, thus avoiding switching to DBN models with discrete and continuous variables and their limiting constraints.

To compare the results obtained with DBN and mtDBN models with another state-of-the-art TS forecasting model, we use long short-term memory (LSTM) neural networks[15] and high-order fuzzy cognitive maps (HFCM).[16] LSTM models have found much use and popularity in recent years,[17] and they have been applied to a wide range of TS forecasting problems. On the other hand, fuzzy cognitive maps are graph-based TS forecasting models. They share similarities with recurrent neural networks and use fuzzy sets to represent the relationships between the variables in real world problems. They have seen much use in recent decades and many studies have been presented introducing new variations of this framework.[18]

The main contribution of this study is the definition of a hybrid model between model trees and DBNs, which allows nonlinearity in the forecasting via piecewise regression. Our results in three different data sets show that the mtDBN model outperforms DBN models and is competitive with other state-of-the-art TS forecasting models. The mtDBN model learning and inference are also distributed as an open source R package to facilitate its future possible applications.

The rest of the paper is organized as follows. In Section 2, we introduce the background of DBN models. Section 3 defines our proposed mtDBN model. In Section 4 we explain the experimental methods and compare the results of the mtDBN and state-of-the-art methods. Finally, in Section 5, we give some final remarks and conclusions.

2 | DBNs

Bayesian networks (BNs)[10] are probabilistic graphical models that represent conditional dependence relationships between variables using a directed acyclic graph (DAG). Each of the nodes in the graph corresponds to a variable in the original system, and the arcs represent their probabilistic relationships. Depending on the type of variables, in the majority of cases these nodes define either multinomial distributions or Gaussian distributions. This leads to three popular different types of models: discrete BNs, Gaussian BNs and conditional linear Gaussian BNs, with the latter being used for mixed types of variables. Each of these models also has particular learning and inference algorithms. In this study, we will focus on Gaussian BNs.

2.1 | Gaussian BNs

A Gaussian BN represents a joint distribution $p(\mathbf{X})$ factorized as

$$p(\mathbf{X}) = \prod_{i=1}^{n} p(X_i|\mathbf{Pa}_i), \tag{1}$$

where $\mathbf{X} = \{X_1, ..., X_n\}$ is the set of all nodes in the network and $\mathbf{Pa}_i = \{X_{1(i)}, ..., X_{k(i)}\}$ is the set of parent nodes of node X_i in the graph. Given that we are in the Gaussian scenario, $p(\mathbf{X})$ is a

multivariate Gaussian distribution, and the local probability density of each node in Equation (1) is defined by a conditional probability distribution (CPD) given its parents:

$$p(x_i|\mathbf{Pa}_i) = \mathcal{N}\left(\beta_{0i} + \beta_{1i}x_{1(i)} + ... + \beta_{ki}x_{k(i)}; \sigma_i^2\right) = \mathcal{N}\left(\mu_i^{\mathbf{Pa}_i}; \sigma_i^2\right). \tag{2}$$

In Equation (2), $\beta_{0i}, ..., \beta_{ki}$ are the regression parameters associated with each parent node in $\mathbf{Pa}_i$, and σ_i^2 is the unconditional variance of X_i that does not depend on its parents. When all the nodes in the network have this kind of CPD, according to Equation (1) the joint probability distribution of the network can be written as

$$p(\mathbf{X}) = \prod_{i=1}^{n} \mathcal{N}\left(\mu_i^{\mathbf{Pa}_i}; \sigma_i^2\right). \tag{3}$$

An advantage that this model provides is that it can be transformed into its multivariate Gaussian equivalent form:

$$\mathcal{N}(\boldsymbol{\mu}; \boldsymbol{\Sigma}) = \mathcal{N}\left(\begin{bmatrix}\mu_1 \\ \vdots \\ \mu_n\end{bmatrix}, \begin{bmatrix}\sigma_1^2 & \cdots & \sigma_{1n} \\ \vdots & \ddots & \vdots \\ \sigma_{n1} & \cdots & \sigma_n^2\end{bmatrix}\right), \tag{4}$$

where $\mu_1, ..., \mu_n$ are the marginal means of the nodes, σ_i^2 is the variance of node X_i, as in Equation (2), and σ_{ij} is the covariance of nodes X_i and X_j. The form in Equation (4) is very useful to perform fast inference with the multivariate Gaussian equivalent instead of using the marginal and conditional distributions of the Gaussian BN. This is a specific advantage of Gaussian BNs over other BN models, where inference can be very time consuming.

2.2 | Dynamic extension

To apply Gaussian BNs to TS data, we need to extend them to DBNs so that they can take time into account. This extension discretizes time into consecutive time slices that influence each other. Each time slice can have a local structure, whose arcs we refer to as intra-slice, and can be connected with previous and posterior time slices with arcs that we denominate inter-slice. Inter-slice arcs can only go from previous time slices to more recent ones, so they pose no problem to the DAG condition of BNs, as they cannot introduce cycles. Inter-slice arcs represent the effects of the past in the current state of the system.

To calculate the joint probability distribution, we now take into account all previous time slices up to the horizon T we want to predict:

$$p(\mathbf{X}^0, ..., \mathbf{X}^T) \equiv p(\mathbf{X}^{0:T}) = p(\mathbf{X}^0) \prod_{t=0}^{T-1} p(\mathbf{X}^{t+1}|\mathbf{X}^{0:t}), \tag{5}$$

where $\mathbf{X}^t = \{X_1^t, X_2^t, ..., X_n^t\}$ is the set of nodes at time slice t. However, DBNs usually assume that the past only affects the present up to a certain order. We call this the Markovian order of

the network, and it simplifies the DBN model greatly. With this assumption in place, the joint probability distribution of the network can be represented as

$$p(\mathbf{X}^{0:T}) = p(\mathbf{X}^0) \prod_{t=0}^{m-1} p(\mathbf{X}^{t+1}|\mathbf{X}^{0:t}) \prod_{t=m}^{T-1} p(\mathbf{X}^{t+1}|\mathbf{X}^{(t-m+1):t}), \tag{6}$$

where m is the Markovian order of the network. A very common assumption is to set the Markovian order to 1 so that only the last instant affects the next state of the system. This further simplifies the general formula in Equation (6) to the specific case of $m = 1$:

$$p(\mathbf{X}^{0:T}) = p(\mathbf{X}^0) \prod_{t=0}^{T-1} p(\mathbf{X}^{t+1}|\mathbf{X}^t). \tag{7}$$

The assumption in Equation (7) can be relaxed to allow inter-slice arcs from older time slices when the autoregressive order of the TS we are treating indicates to do so. An example of the structure of a DBN with Markovian order 2 is shown in Figure 1.

Another common simplifying assumption is to avoid having a local structure in each time slice and allow only inter-slice arcs. This dilutes the effect that the variables have on other variables in the same instant, but in return makes it easier to avoid cycles in the creation of network structures and results in sparser graphs.

One advantage that DBN models present is that they do not need to be trained with TS of the same length. Due to the Markovian order assumption in Equation (6), we only need to recover several batches of $m + 1$ size from the original TS to learn the structure and parameters of the network. We can use several unequal length TS recovered from the same stochastic

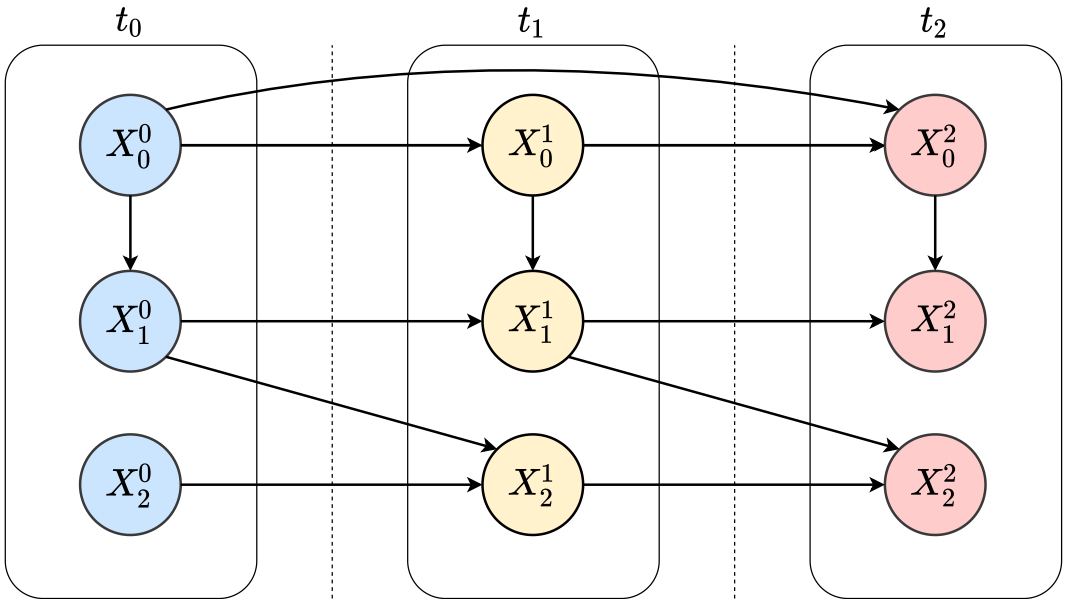

FIGURE 1 Example of the structure of a DBN with three time slices t_0, t_1 and t_2 and Markovian order 2 showing both its inter- and intra-slice arcs. DBN, dynamic Bayesian network. [Color figure can be viewed at wileyonlinelibrary.com]

process to train a DBN model from data. The reason for this is that we only need the values of the variables inside the temporal window defined by the Markovian order to train our model, so the total length of the TS is not relevant in the learning phase. This helps when applying this kind of model to real-world problems, where the length of the data from industrial processes can vary depending on circumstances outside of the system.

3 | TREE-BASED DYNAMIC BAYESIAN NETWORKS

3.1 | Hybrid definition: mtDBN

Our proposed mtDBN model consists of a hybrid between model trees[13] and DBNs in a similar fashion to Bayesian multinets[19] in a dynamic scenario. The objective of the tree is to divide instances from the original data set into different contexts leading to each leaf node. Afterwards, we train several DBN models using the different instances attached to each leaf node. Similar to a multinet architecture, all the networks in the leaf nodes are related to each other, given that they model different situations from the same process, and the tree structure serves as a switch that selects the appropriate model for some new instance that we want to forecast. We chose model trees as the switch in our hybrid because it allows the DBN model to retain its interpretability. The leaf nodes in a tree are clearly differentiated by each branch split, and one can easily see the characteristics of the instances in each leaf node. This way, we can evaluate the specific DBN fitted to a leaf node knowing the context of that leaf node.

Typically, model trees require a response variable to be predicted, although in DBN models any variable can be the objective of inference. Thus, depending on the problem, we can choose any variable in the system as the objective variable for both growing the tree structure and forecasting in the network. Moreover, there is also the possibility of growing a multivariate regression tree[20] and performing inference over several variables at the same time with multi-output regression.

Similar to a model tree, we may have a Gaussian DBN model in each leaf node instead of a linear regression. Given a data set $\mathbf{D}$, we can fit either a univariate or multivariate tree model and then evaluate which leaf node each of the instances belongs to. With these subdatasets, we can train each of the aforementioned DBNs and store them in a vector $\mathbf{M}$ of models and link each leaf node with one position of this vector. Then, while performing forecasting, a new instance is sorted down the tree to find its corresponding leaf node, and then we use the network inside $\mathbf{M}$ that is attached to that leaf node to forecast. Afterwards, if we are predicting up to a certain horizon in the future, the results of the forecasting are processed again with the tree to select a new DBN model to continue the forecasting. This way, we select an appropriate network based on the state of the system at each instant. Given that the inference performed by Gaussian DBNs is linear, with this hybrid we can augment it to perform piecewise linear regression to achieve pseudo nonlinear forecasting with DBN models. The architecture of the hybrid mtDBN model is depicted in Figure 2.

In our case, we opted to use a CART model[12] with reduced variance splitting criterion for the tree structure that classifies instances into different leaf nodes. We chose this model due to its generality and simplicity as a starting point for the hybrid. CART models are well established and extended, and variance reduction splitting can be applied as a general method for continuous data. The model that defines the multinet, in our case the CART model, can potentially be swapped for more appropriate ones in different applications, such as a clustering method or another type of tree-based model. The splitting criterion can also be defined differently to prioritize specific

behaviours, for example, using information gain[21] or change point detection methods on TS.[1] In this first version, we want to establish a more general approach that can be specialized to different scenarios. The pseudocode of the learning phase of our hybrid model is shown in Algorithm 1. We will refer to specific lines during the remainder of this section.

Algorithm 1: Learning the mtDBN model from data

Input: Parameters α, inc_cte, min_inst, $homogen$, mv
Output: The hybrid model $\mathbf{M}_{mtDBN}$
Data: Training set $\mathbf{D}$

```
1  valid = False;
2  full_tree = growCART(D, α, mv);
3  while not valid do
4      pruned_tree = pruneCART(full_tree, α);
5      D_class = classifyDataWithTree(D, pruned_tree);
6      valid = checkInstancesPerLeafNode(D_class, min_inst);
7      α = α + inc_cte
8  if homogen then
9      dbn_struc = learnDBNStruct(D);
10 M_mtDBN = [ ]
11 for unique tree node data D_leaf in D_class do
12     if not homogen then
13         dbn_struc = learnDBNStruct(D_leaf);
14     M^i_mtDBN = fitDBNParameters(D_leaf, dbn_struc);
15 return M_mtDBN;
```

Our data set $\mathbf{D}$ contains all instances of our regressor variables X_j inside a table structure:

$$\mathbf{D} = [\mathbf{X}_0, ..., \mathbf{X}_n] = \begin{bmatrix} x_{00} & \cdots & x_{n0} \\ \vdots & \ddots & \vdots \\ x_{0l} & \cdots & x_{nl} \end{bmatrix}, \tag{8}$$

where l denotes the total length of the data set. The specific value of variable X_j in instance k is thus defined by x_{jk}. In Algorithm 1, we provide $\mathbf{D}$ as input data.

We begin by defining an objective response variable to grow the tree. In our implementation, we allow both univariate and multivariate model trees, but in this section, we focus on univariate trees for simplicity. Note that, no matter the kind of model tree used, the mtDBN model will always perform multivariate inference. Once we set the response variable Y, which is one of our X_j variables, we grow the tree defining cut-offs on some regressor variable X_j based on the sum of squares of Y:

$$SS = \sum_i (y_i - \bar{y})^2, \tag{9}$$

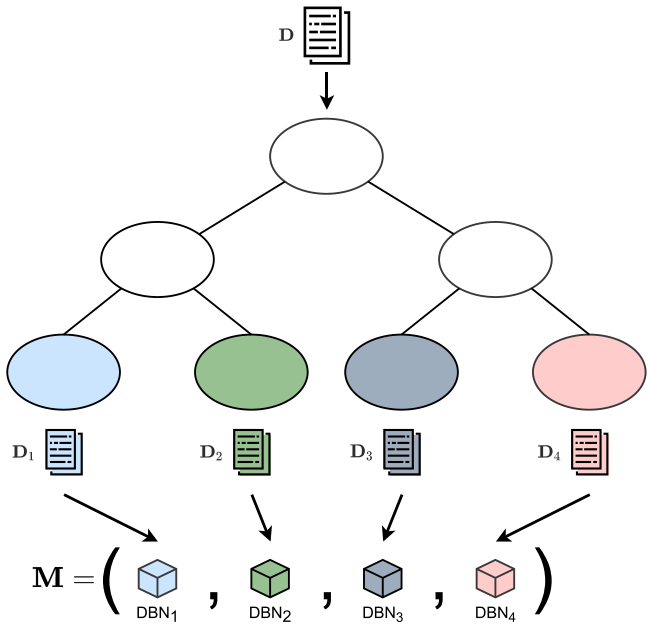

FIGURE 2 Schematic representation of the mtDBN architecture. The tree structure is initially used to divide the original data set **D** into several data sets $\mathbf{D}_{leaf}$. Each one of these data sets is then used to fit a DBN model, which will be stored in the model vector **M**. Afterwards, the tree structure will be used to classify new instances according to the appropriate DBN model inside **M**. mtDBN, model tree dynamic Bayesian network. [Color figure can be viewed at wileyonlinelibrary.com]

where y_i is the value of Y in an instance of our data set and $\bar{y}$ is the sample mean. The only difference with multivariate trees is that this sum of squares would be calculated for a vector **Y** of response variables instead of a single response variable. We calculate the cut-off point that generates two new tree nodes by maximizing the reduction of SS on both branches rooted at X_j compared to the parent node:

$$\delta_{tree} = SS_p - (SS_l + SS_r), \tag{10}$$

where δ_{tree} is the total improvement of a split, SS_p is the sum of squares of the parent tree node, SS_l is the sum of squares of the left branch and SS_r is the sum of squares of the right branch. We calculate the best δ_{tree} for all X_j variables available and then choose the one that maximizes it. Given that our regressor variables are continuous, there are potentially infinite possible cut-off points. To solve this issue, we only check the cut-off points defined by the values of X_j in our data D; that is, we evaluate Equation (10) for $X_j \geq x_{jk}$ in the left branch and $X_j < x_{jk}$ in the right branch. During the growing phase, we grow a univariate or multivariate tree until either none of the possible new splits obtain a positive δ_{tree} or we reach a maximum tree depth defined by the user in line 2 of Algorithm 1. The input parameters *homogen* and *mv* in Algorithm 1 are used to determine whether a homogeneous or nonhomogeneous and univariate or multivariate tree is built. Once we grow the tree, we prune it by fixing an $\alpha \in (0, 1]$ parameter that defines a threshold over δ_{tree}. If a split does not satisfy $\delta_{tree} \geq \alpha\delta^P_{tree}$, where δ^P_{tree} is the total improvement of the previous tree node, then that subtree is pruned. The growing and pruning of the tree is encapsulated from line 2 to line 7 of Algorithm 1. In addition, we want to focus some attention

on line 5 because it labels each row in $\mathbf{D}$ to its corresponding leaf node in the pruned tree and generates the $\mathbf{D}_{class}$ dataset. This will be key in the next steps of the algorithm when we use specific instances to learn each DBN model.

It is important to note that we apply the splitting criterion to the TS data in a static fashion, without taking into account the time difference between instances. This is because we want to be able to select the appropriate DBN model based on the current state of the system in each instant. If one wants to add the temporal component to the tree construction, the time passed since the beginning of the process may be added as a variable. This could potentially give the tree an idea of how long the system has been running, and it allows splits based on how much time has passed.

When we obtain an adequate tree model, we use the splitting rules defined by the tree to process the original dataset and assign each instance to its corresponding leaf node, generating several subdatasets. Afterwards, we use these subdatasets to fit a different DBN model for each leaf node. This way, each of the DBNs is tuned to its specific context as defined by the tree structure. This process is shown in Algorithm 1 from line 8 to line 14. These models can be homogeneous if they all share the same network structure learned from the whole dataset in lines 8 and 9, but each has different parameters fitted in line 14, or nonhomogeneous if both the structure and the parameters of each network are learned from the local instances of each leaf node in lines 12 to 14. The loop in line 11 iterates over the different subdatasets $\mathbf{D}_{leaf}$ generated by the classification in line 5. Each $\mathbf{D}_{leaf}$ contains only those instances corresponding to a specific leaf node to fit the DBN with that specific part of the complete data set $\mathbf{D}$.

As a reliability measure, we added the *min_inst* parameter that sets a minimum number of instances per leaf node. This avoids training DBN models with few data. We perform this by pruning the tree structure and checking the number of instances per leaf node with that structure. If it has fewer instances than the minimum allowed, we increase the parameter α by a constant defined by the user with the *inc_cte* parameter and repeat the pruning process to generate shallower trees from line 5 to line 7. In particular, in line 6, we declare a pruned tree valid or invalid by recounting the number of instances in each leaf inside $\mathbf{D}_{class}$ and checking that they are all above the minimum.

3.2 | Forecasting

Once a tree is obtained and all necessary DBN models are fitted, they can be used simultaneously to perform forecasting over some time horizon T. We begin the forecasting procedure with an initial evidence vector $\mathbf{s}_0$ defined as

$$\mathbf{s}_0 = (\mathbf{x}^0, \mathbf{x}^1, ..., \mathbf{x}^t) = \left(\left(x_1^0, x_2^0, ..., x_n^0\right), ..., \left(x_1^t, x_2^t, ..., x_n^t\right)\right), \tag{11}$$

where n is the number of variables per time slice and $t + 1$ is the number of time slices in the network. This vector $\mathbf{s}_0$ defines the state of the system at the initial point of the forecast. The t-th time slice represents the variables in the present, and only $(x_1^t, ..., x_n^t)$ will be processed by the tree structure to determine which of the DBN models in the mtDBN will be the most suitable to forecast the next instant. Inspired by model trees, we also fit a DBN model on the tree nodes just before the leaf nodes. When we perform the forecast, we use both the

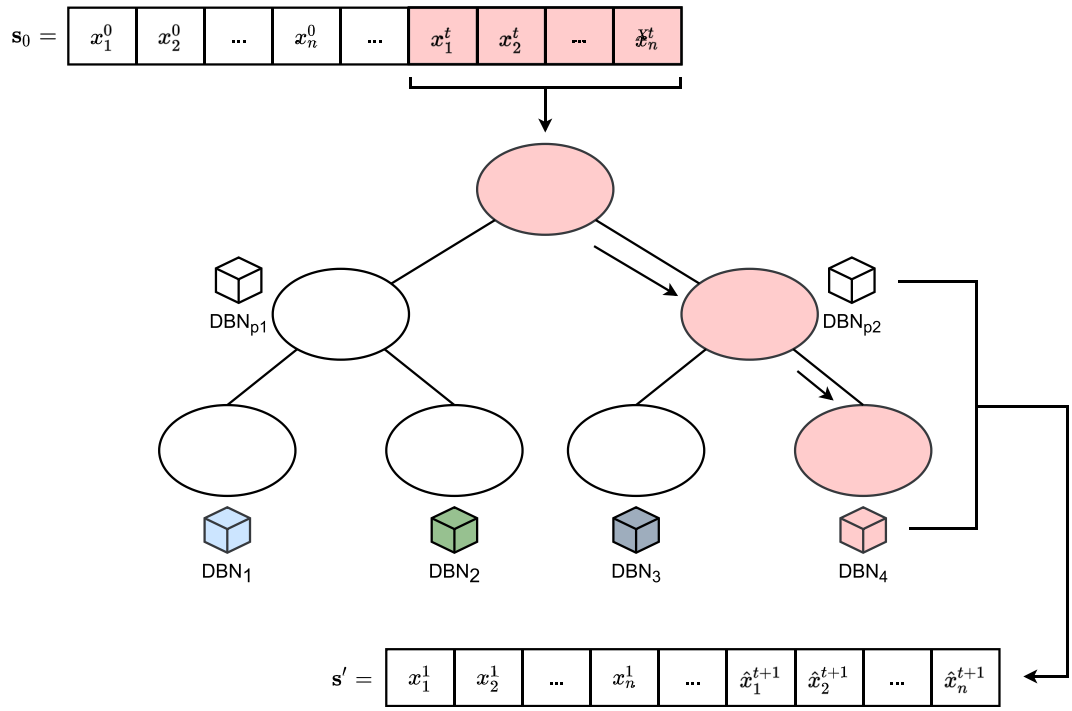

FIGURE 3 Representation of the mtDBN forecasting a single instant. The state vector is classified by the tree, and the correspondent DBN leaf and parent models are used to predict the next state vector. DBN, dynamic Bayesian network; mtDBN, model tree dynamic Bayesian network. [Color figure can be viewed at wileyonlinelibrary.com]

correspondent DBN model attached to the leaf node of the tree and the DBN attached to the parent node of that leaf. Afterwards, we average the forecasting results of both DBN models to obtain a final forecast of the next state of the system. This step helps the hybrid model obtain smoother transitions between the DBNs of each leaf node. Once we obtain the new forecasted vector $\hat{\mathbf{x}}^{t+1} = (\hat{x}_1^{t+1}, \hat{x}_2^{t+1}, ..., \hat{x}_n^{t+1})$ from the inference of the two DBN models, we delete the oldest $\mathbf{x}^0$ and insert $\hat{\mathbf{x}}^{t+1}$ as the new $\mathbf{x}^t$, moving all the evidence for the remaining time slices forward in a sliding window fashion. The whole process is illustrated in Figure 3.

The new $\mathbf{s}'$ vector obtained after all the evidence is moved forward is used in the next forecasting step and processed by the tree before the next forecast. This way, when forecasting with our hybrid model to some horizon, a DBN will be dynamically selected in each forecasting step based on the current state of the system. This provides the desired nonlinear behaviour with piecewise forecasts and allows the model to adapt to both nonlinear systems and sudden interventions or drifts.

4 | EXPERIMENTAL RESULTS

To test the effectiveness of our proposed model, we set up several experiments with both synthetic and real-world data from nonlinear processes. By analysing the forecasting results, we can assess how the mtDBN model fits nonlinear processes in comparison with classical DBN models, LSTM neural networks and HFCMs, which are well behaving and popular models.

All the programming code of the models, experiments and data sets can be found online in a GitHub repository (https://github.com/dkesada/mtDBN). The DBN and mtDBN models are written in both R and C++, the latter used for expensive computations, and distributed as R packages on CRAN and GitHub. All experiments were performed on an Ubuntu 18 machine with an Intel i5-4790k processor and 16 GB RAM.

4.1 | Simulated nonlinear problem: Fouling phenomenon

To obtain performance results in a controlled environment, for our first experiment we generated a synthetic data set from a nonlinear process defined by a system of ordinary differential equations (ODEs). Our scenario defines a chemical reaction that occurs inside industrial furnaces. As we heat a chemically reactive fluid to some desired temperature, this fluid can be prone to depositing solidified impurities on the inside of the pipes. As time passes, these impurities can generate an insulating layer that will force us to progressively increase the furnace temperature to keep the fluid temperature inside the pipes constant. This process is called *fouling*, and it is the cause of severe costs in terms of efficiency loss.[22] Due to the inherently nonlinear nature of the physical relationships that define this phenomenon, it can be used as an example to test the effectiveness of our hybrid model.

To generate our data set, we used a system of ODEs that define a simplified fouling phenomenon without spatial dimensions. We modelled the following 10 variables: the fluid temperature T_1, the tube wall temperature T_2, the thickness of the fouling layer S_c, the concentration C_a of particles in the fluid prone to depositing, the amount of fuel m_c administered to the furnace heaters, the density ρ_1 of the fluid and its thermal capacity C_{p1}, the flow inside the tubes Q_{in}, the volume *vol* and concentration of particles C_{ain} in the fluid prone to depositing that is renewed at each instant. The details about the simulation construction and definition are further discussed in Appendix A.

Once the simulation was created, we generated independent and identically distributed multivariate TS of 100 time instants running a cycle. Each cycle represents the case where the furnace is reverted to an initial state after cleaning the fouling layer inside the tube, and we performed 100 time instants of operation as the time horizon. For each cycle, we set random starting values for the fluid properties and several different behaviours for the temperature and flow variations so that different fluids and furnace configurations were seen in the data. In total, we generated a data set of 1000 cycles of 100 time instants each, that is, 100,000 instances and 10 variables. Then, we performed a 30-fold cross-validation where 967 cycles were used in training and 33 were used for testing in each of the 30 hold-out processes. In our experiments, we used the first instance of each cycle as evidence, and we forecast T_1 until the last time instant. The mean absolute error (MAE) and mean absolute percentage error (MAPE) were estimated in all instants and then averaged in all cycles to obtain the accuracy of each model. The DBN structures were learned using a particle swarm algorithm.[23]

To assess our proposed model, we compare mtDBN variations that employ multivariate and univariate trees and homogeneous and nonhomogeneous DBN structures for the models attached to the leaf nodes. Univariate trees can be better suited for problems where we are only interested in forecasting one variable. With our DBN models, we always forecast the complete multivariate state vector, but the tree structure can focus on a single objective variable that is of interest. With multivariate trees, we can cover situations where we may have several objective variables that we are interested in forecasting. On the other hand, nonhomogeneous models

can perform better when the process that we are trying to model is nonstationary, given that the conditional independence relationships defined by the DBN structures can differ from one leaf model to another. Homogeneous models are expected to work better in modelling processes that are nonlinear in nature but do not drift over time because the dependence between variables will always have the same structures but different parameters underneath. We set the Markovian order of the DBNs to 1, given that we know beforehand that our ODE system is autoregressive of order 1 and only uses the last time instant to generate the next one.

We also tune and fit LSTM and HFCM models in the experiment for the sake of comparing the performance of DBN-based models with other state-of-the-art models for TS forecasting. LSTM models are specifically tailored to forecast both univariate and multivariate TS, and their recurrent structure can be used to perform forecasts of arbitrary length, similar to the case of DBN models. The code of these models and the experiments can also be found online in a GitHub repository (https://github.com/dkesada/kTsnn). HFCM are also graphical models that can be used for multivariate TS forecasting, and serve as a middle ground between DBNs and LSTMs because they share characteristics with both models. Our version of the HFCM comes originally from a project (https://github.com/julzerinos/python-fuzzy-cognitve-maps) that applies HFCMs to the uWave data set, which is a data set used to train gesture recognizers. An autoregressive version of HFCMs has been programmed by using a sliding window that transforms the predictions of the model into inputs for the next instant. This was necessary to perform the long-term and midterm forecasting in two of the experiments. The code of both the model and the experiments can also be found online in a GitHub repository (https://github.com/dkesada/HFCM).

The results in Table 1 show that the mtDBN models obtain better MAE and MAPE results than the baseline DBN model at the cost of longer training and execution times. The MAE result of the best hybrid model is almost half the baseline MAE of the regular DBN model. In the case of nonhomogeneous models, where each leaf node has a different DBN structure, the training time increases due to having to run the structure learning algorithm for all the different contexts. In this experiment, we can see that the model that excels is the homogeneous and multivariate mtDBN. This can be explained from the point of view of the process that we are modelling. Our simulation generates traces from a nonlinear system of ODEs that does not change over time. This means that the relationships established between the variables remain constant throughout,

TABLE 1 Results in terms of the MAE, MAPE, training and execution time of the models for the fouling experiment.

	Homogen?	Tree	MAE	MAPE	Train (m)	Exec (s)
DBN	–	–	80.23	13.42	7.99	0.58
mtDBN	Yes	Univariate	60.33	9.72	8.08	1.388
mtDBN	Yes	Multivariate	49.60	8.07	8.29	1.387
mtDBN	No	Univariate	79.52	12.83	32.91	1.388
mtDBN	No	Multivariate	54.15	8.90	32.04	1.387
LSTM	–	–	59.81	10.20	7.09	0.08
HFCM	–	–	177.32	27.29	59.27	1e-3

Abbreviations: DBN, dynamic Bayesian network; HFCM, high-order fuzzy cognitive maps; LSTM, long short-term memory; MAE, mean absolute error; MAPE, mean absolute percentage error; mtDBN, model tree dynamic Bayesian network.

and this is best represented by homogeneous models. Multivariate trees also seem to outperform their univariate counterparts, likely because the DBN models perform multivariate inference themselves. The execution time in the mtDBNs is consistently higher due to having to classify the state of the system in each forecasting step with the tree structure to select the appropriate DBN model and having to forecast with both the leaf node model and the parent node model to smooth the results. However, given that the forecasting process of all the mtDBN models is the same, all their execution times are equivalent. When we compare the best resulting mtDBN model with the baseline unique DBN, we can see that we do not sacrifice too much training and execution time in exchange for a significant reduction in the MAE and MAPE.

Given that our proposed mtDBN model combines DBN models with model trees, we want to ensure that the results obtained with the hybrid are statistically significant when compared with the baseline model. For this purpose, we performed pairwise Wilcoxon rank-sum tests to check that the mean accuracies obtained from the mtDBN models are statistically significantly better in comparison with the baseline model. We chose this non-parametric test because the samples do not follow Gaussian distributions. The results shown in Table 2 show that all mtDBN models reject the null hypothesis of equal performance in MAE. We also performed this test between the baseline model, the best mtDBN model and the LSTM model with multiple comparisons and observed from the *p*-values that they all reject the null hypothesis of equal performance in MAE, as shown in Table 3. Given that we have more than two data samples from all the tested models, we also performed the Kruskal-Wallis test on our results. This allows us to perform a non-parametric test that does not rely on pair wise tests between the models. The test obtains a *p*-value of 2.2e-16, which indicates that at least one of the sample results from the models is better and statistically significant from the others.

TABLE 2 Resulting *p*-value of the Wilcoxon rank-sum tests for the forecasts of the different hybrid models in comparison with the baseline DBN model for the fouling experiment.

Homogen?	Tree	*p*-value
Yes	Univariate	2.200e−16
Yes	Multivariate	2.200e−16
No	Univariate	1.471e−11
No	Multivariate	2.200e−16

Abbreviation: DBN, dynamic Bayesian network.

TABLE 3 Resulting *p*-value of the Wilcoxon rank-sum tests in the multiple comparisons between the baseline DBN model, the homogeneous multivariate mtDBN, the LSTM and the HFCM model.

	DBN	mtDBN	LSTM	HFCM
DBN	–	2.20e−16	1.82e−08	2.20e−16
mtDBN	2.20e−16	–	2.20e−16	2.20e−16
LSTM	1.82e−08	2.20e−16	–	2.20e−16
HFCM	2.20e−16	2.20e−16	2.20e−16	–

Note: All the tests reject the null hypothesis of equal performance in MAE.

Abbreviations: DBN, dynamic Bayesian network; HFCM, high-order fuzzy cognitive maps; LSTM, long short-term memory; mtDBN, model tree dynamic Bayesian network.

In comparison with the LSTM model, the MAE results are much better than the baseline and similar to the hybrid models. In terms of time costs, training times of the LSTM model are similar to the DBN and homogeneous mtDBN models, and its execution time is the second best, only behind the HFCM model. They are very powerful models in this kind of settings, and their only remarkable downsides are that they are black box models and that the tuning process can be very time consuming due to the large number of parameters and network architectures that may be evaluated before finding some optimal configuration being guided blindly. In contrast, the HFCM model works noticeably worse than the rest of the models in this synthetic long-term case. This is due to the fact that fuzzy cognitive map models perform well on short-term predictions, but worse on long-term predictions.[24] The training time ramps up due to the internal optimization of the weight matrix having to work with long cycles of 100 instances each, and the accuracy of the model is almost twice as worse than the baseline DBN model. In terms of execution time though, it is by far the fastest model.

Figure 4 shows an example of the tree structure from an mtDBN model. This tree structure is interpretable and can garner valuable insights into the problem at hand. For example, in this particular case, we know that the amount of fuel m_c administered to the furnace is an external parameter decided by a human operator. Higher m_c on the right branch of the tree generates higher

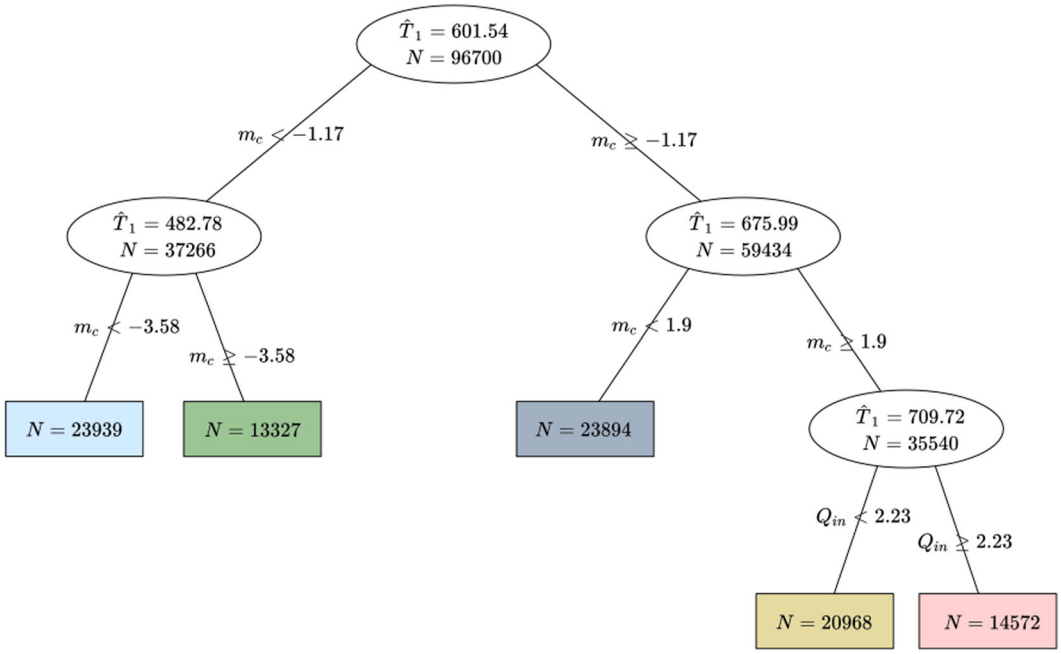

FIGURE 4 Example of the tree structure of a homogeneous multivariate mtDBN model. The resulting tree has five different leaf DBN nodes, represented in various colours. The splitting rules are shown in the branches, and the average value of the objective temperature T_1 and the number of instances N per node are shown inside the tree nodes from the initial 96,700 instances in the training data set at the root of the tree. From the tree, we can identify that the m_c variable reduces the sum of squares the most in the different scenarios. From the root node, lower quantities of fuel m_c administered to the furnace will result in slower processes with lower T_1 and vice versa. In the case of extremely high T_1, the most severe cases are defined by a high flow of fluid Q_{in} entering the system. DBN, dynamic Bayesian network; HFCM, high-order fuzzy cognitive maps; LSTM, long short-term memory; mtDBN, model tree dynamic Bayesian network. [Color figure can be viewed at wileyonlinelibrary.com]

fluid temperatures T_1 and promotes a faster creation of the insulating fouling layer S_c. This increase in S_c means that increasingly higher values of m_c will be needed to maintain a high T_1. Additionally, the reaction is further characterized by the flow rate Q_{in} on the rightmost branch with the highest T_1 cases. A fast flow rate can only be coupled with high T_1 if the furnace is near its maximum temperature and the process has not been running for long, because it is very unlikely to achieve a high T_1 along with a fast flow rate and an already thick insulating layer. This differentiation with S_c of whether the furnace is working at different capacities improves the overall performance of the mtDBN compared to the baseline DBN by allowing the hybrid to switch between models in the middle of forecasting based on the current state of the system. It is important to note that we can explain the cuts performed by the tree structure more in-depth because we already know the underlying process that we are modelling. In a real case scenario, the tree structure offers valuable information on how this process operates, but we will likely not reach the level of interpretation shown in this example without the insight of an expert on the problem.

For illustration purposes, we also interpret the structure of a DBN model in the homogeneous multivariate mtDBN in Figure 5. We see how the T_1 and T_2 values at the previous instant affect the predictions of the current value of T_1, which is directly defined in the underlying system of equations. Both m_c and Q_{in} affect the objective temperature, as hinted at

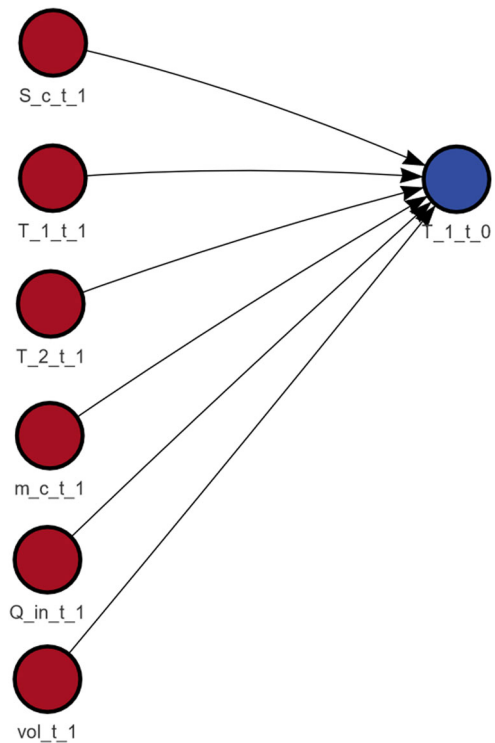

FIGURE 5 Example of the DBN structure of the homogeneous multivariate mtDBN model. The objective variable T_1 at the present instant is represented as the blue node, and all the parent nodes in red represent those variables at the previous instant. With these parent nodes, we can see the autoregressive component and the most relevant variables in the inference of T_1, that is, S_c, T_2, m_c, Q_{in} and *vol*. DBN, dynamic Bayesian network; HFCM, high-order fuzzy cognitive maps; LSTM, long short-term memory; mtDBN, model tree dynamic Bayesian network. [Color figure can be viewed at wileyonlinelibrary.com]

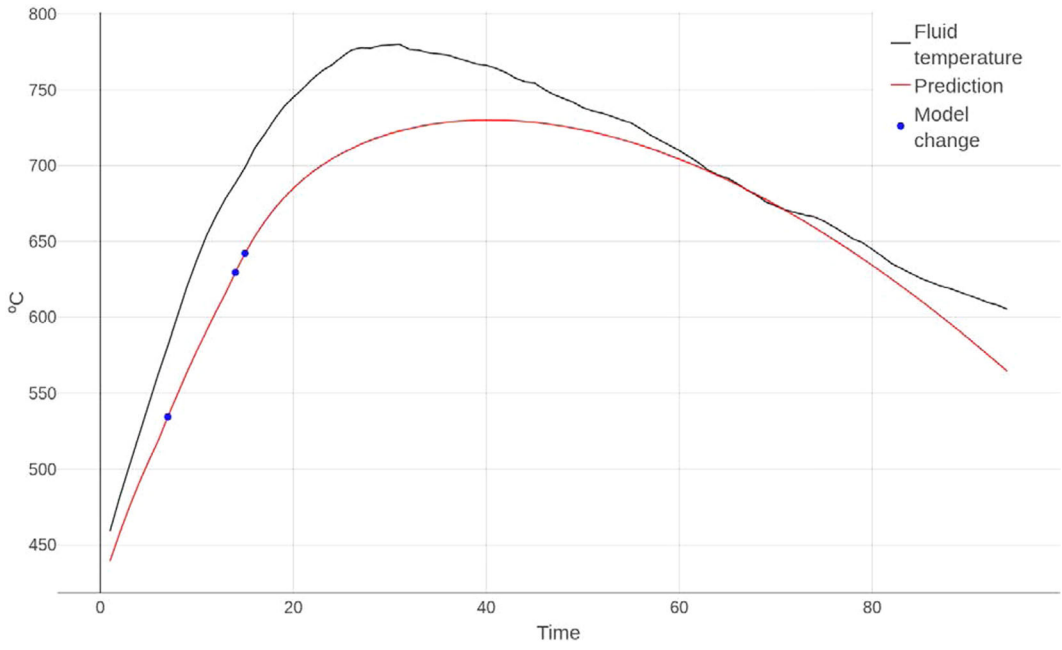

FIGURE 6 Example of predicting the fluid temperature T_1 of a 100 instant trace using the homogeneous multivariate mtDBN. The model used for forecasting is changed three times during forecasting based on the predicted state of the system at each instant. It is important to note that we only show the fluid temperature in the figure, but all variables in the system are jointly forecasted simultaneously to obtain the state of the system at the next instant. mtDBN, model tree dynamic Bayesian network. [Color figure can be viewed at wileyonlinelibrary.com]

by the tree structure, and finally, both S_c and the volume are also relevant to the variation in T_1. By using both the tree and DBN structures, we can identify the key variables that intervene in the estimation of our objective variable T_1 in the underlying process, and we can check the specific effect that these variables have in the inference performed by the DBN.

To put the profile of the forecasted temperature curves into context and to illustrate what the predictions look like, an example of forecasting a cycle with the best mtDBN model is shown in Figure 6. In total, four different leaf node models are used in that specific forecasting to obtain the desired piecewise prediction of the temperature.

4.2 | Real data application: Electrical motor

To observe the effectiveness of our hybrid model using real world-data, we set up a similar exercise using public experimental data collected from an electrical motor.[25] The motor speed, temperature in different sections, voltages and torque were recovered each instant from sensors, generating a multivariate TS data set of several recording sessions with a total of 11 variables. The objective is to forecast the rotor temperature of the motor, given that this feature is not easily monitored inside electric motors and obtaining accurate predictions can increase the efficiency due to being able to predict overheating situations ahead of time and prevent them.

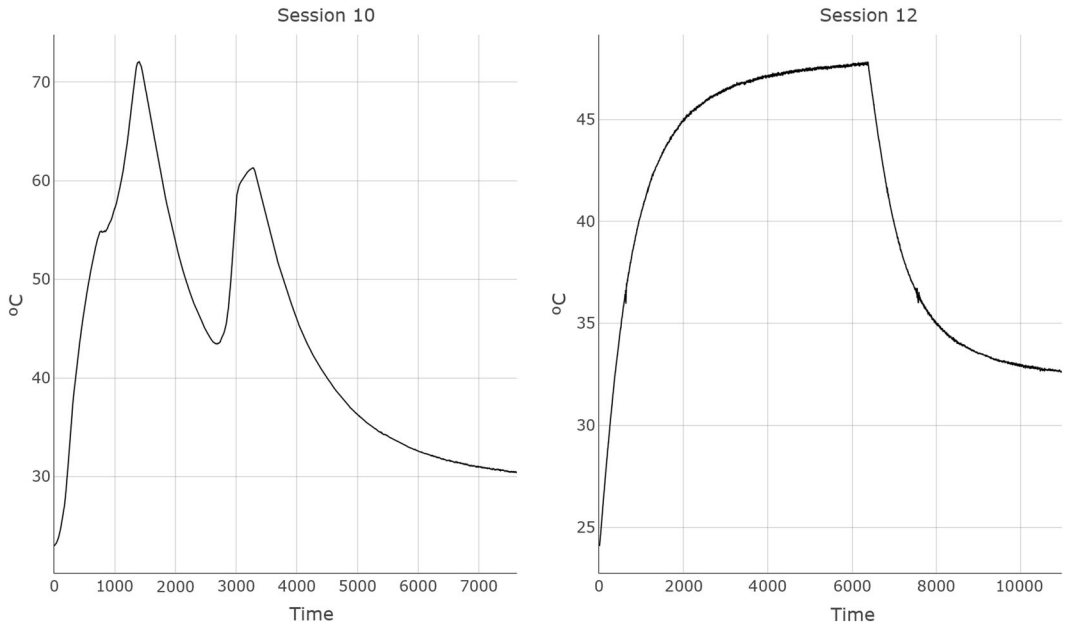

FIGURE 7 Example of the objective temperature in two different sessions in our data set. Session 10 is rather irregular, with several interventions drastically changing the tendency of the series. Session 12 is a more common case in our data, with only one significant intervention after 6000 s.

The data set consists of 69 different recording sessions. A session consists of recordings of the active motor from an initial idle state up to some point in time. The values of the sensors were recorded with a frequency of 2 Hz, that is, each row is separated by 0.5 s. There are a total of 1,330,816 instances in the data set, with the shortest session spanning 2176 instances (18.13 min) and the longest 43,971 (6.12 h). In this scenario, we face two problems: high dimensionality in terms of the number of instances and sessions of different lengths in the data. Performing long-term forecasting of complete cycles from a single starting point seems unreasonable due to the length of the TS. To aggravate this problem, the profile of the curves is very irregular, and interventions in the system affect the tendency during sessions, as seen in Figure 7.

To approach the dimensionality issue, we reduce the frequency of the data to $0.0\dot{3}$ Hz so that rows are separated by 30 s each. When reducing the frequency, the mean of each 60-instance bin is returned as the new value. After this process, the data set is reduced to 22,247 instances total, with sessions ranging from 37 to 734 rows. To check that we are not losing too much information with this reduction of frequency, we compute the average distance between the original TS and the reduced ones using the dynamic time warping distance implemented in the *dtw* R package.[26] Given that the TS are not too noisy, as seen in Figure 7, and that we are averaging binwise, we do not expect the reduced versions to be very distant from the original ones. The total average normalized distance between the series is 0.1291, and this coupled with the alignments depicted in Figure 8 indicates that there was not a severe information loss during the frequency reduction step.

To address cycles of different lengths, we forecast up to fixed intervals of time instead of the whole sessions. We perform several consecutive forecasts of 20 instants. In the DBN models, we fixed the Markovian order to 2 for the structure of the networks because this real case scenario

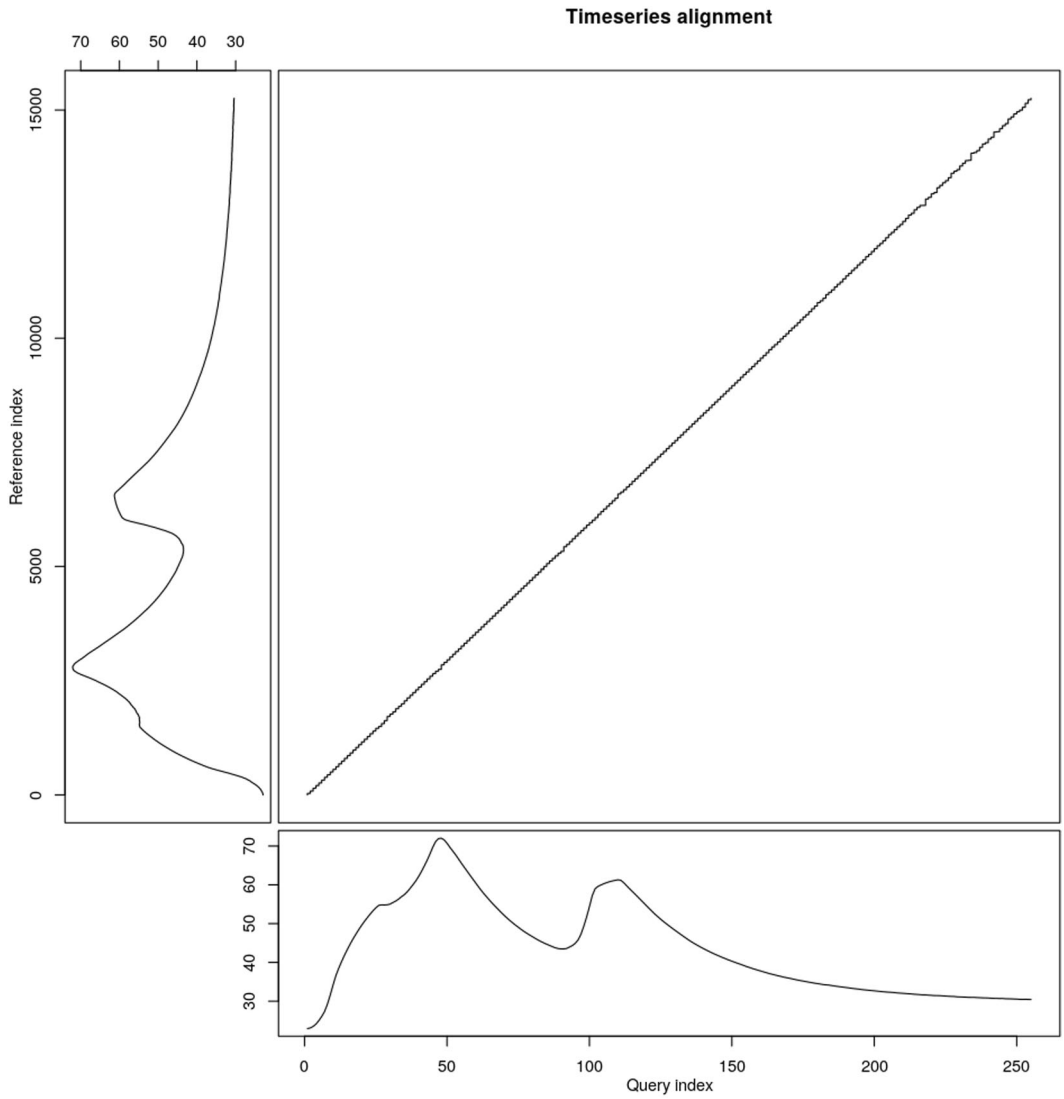

FIGURE 8 Example of the alignment of two TS sessions of the objective variable in the original and reduced data sets. The alignment being almost a straight diagonal line indicates that no displacement in time is seen and that no drastic jumps in values are present. TS, time series.

does not necessarily have an autoregressive order of one. The MAE result of these predictions were calculated, and the total MAE of a model is the global mean value of all the forecasts. To divide the data set into training and test partitions, we used threefold cross-validation to form groups of 66 sessions for training and three sessions for testing. This way, the groups were formed randomly and all cycles appear in the test once.

The results of the experiments show that the mtDBN models also improve the MAE and MAPE results of the baseline DBN model in this experiment, as seen in Table 4. In this scenario of midterm forecasting, the different DBN models are able to fit the contexts defined by the tree better than a global linear DBN model. The execution and training times of the hybrid models present the expected characteristics, taking longer to compute, similar to the synthetic case.

TABLE 4 Results in terms of the MAE, MAPE, training time and execution time of the models for the motor data set.

	Homogen?	Tree	MAE	MAPE	Train (m)	Exec (s)
DBN	–	–	1.616	2.970	8.53	0.102
mtDBN	Yes	Univariate	1.499	2.602	8.73	0.215
mtDBN	Yes	Multivariate	1.574	2.807	8.74	0.214
mtDBN	No	Univariate	1.484	2.558	36.45	0.214
mtDBN	No	Multivariate	1.553	2.806	36.59	0.215
LSTM	–	–	2.289	3.937	2.21	0.083
HFCM	–	–	2.958	5.150	37.07	2e-4

Abbreviations: DBN, dynamic Bayesian network; HFCM, high-order fuzzy cognitive maps; LSTM, long short-term memory; MAE, mean absolute error; MAPE, mean absolute percentage error; mtDBN, model tree dynamic Bayesian network.

TABLE 5 Resulting p-value of the Wilcoxon rank-sum tests for the electric motor data set with respect to the regular DBN model.

Homogen?	Tree	p-value
Yes	Univariate	1.540e−03
Yes	Multivariate	1.633e−03
No	Univariate	1.764e−06
No	Multivariate	2.079e−03

Abbreviation: DBN, dynamic Bayesian network.

The best resulting model was the nonhomogeneous univariate mtDBN, which indicates that the underlying process is probably nonstationary and thus better modelled with different DBN structures, although not by a large margin. The univariate trees obtaining better results than the multivariate trees could be due to defining less significant subsets of data in the leaf nodes for the objective of predicting a single temperature. In this midterm scenario, focusing on forecasting the objective variable with univariate trees can lead to a better accuracy if the error on the prediction of the rest of the variables does not add up. In this case, all models perform better than in the synthetic one, which can be seen in the MAPE being lower in Table 4 than in Table 1. The MAPE indicates that the predictions in this experiment are more accurate than in the previous experiment, which is due to performing shorter term forecastings.

We also performed statistical significance tests in this case. The p-values shown in Table 5 indicate that we reject the null hypothesis of equal performance with respect to the unique DBN model for all mtDBN models. As in the previous experiment, we also performed pairwise Wilcoxon rank-sum tests between the models to check for statistically significant differences in performance in terms of the MAE, and all the models rejected the null hypothesis, which can be seen in Table 6. The Kruskal-Wallis test also obtained a p-value of 2.2e-16, which shows that at least one of the models obtains better, statistically significant results from the others.

The tree structure shown in Figure 9 identifies the two temperatures inside the motor, *stator_tooth* and *stator_yoke*, as those that better differentiate the different contexts of the objective temperature, and in the case of very high temperatures, the electric current i_d helps

TABLE 6 Results of the Wilcoxon rank-sum tests in the multiple comparisons for the motor experiment.

	DBN	mtDBN	LSTM	HFCM
DBN	–	1.764e−06	8.981e−16	2.200e−16
mtDBN	1.764e−06	–	2.200e−16	2.200e−16
LSTM	8.981e−16	2.200e−16	–	1.894e−05
HFCM	2.200e−16	2.200e−16	1.894e−05	–

Note: Similar to the synthetic case, all the pairwise tests reject the null hypothesis of equal performance in MAE.

Abbreviations: DBN, dynamic Bayesian network; HFCM, high-order fuzzy cognitive maps; LSTM, long short-term memory; MAE, mean absolute error; MAPE, mean absolute percentage error; mtDBN, model tree dynamic Bayesian network.

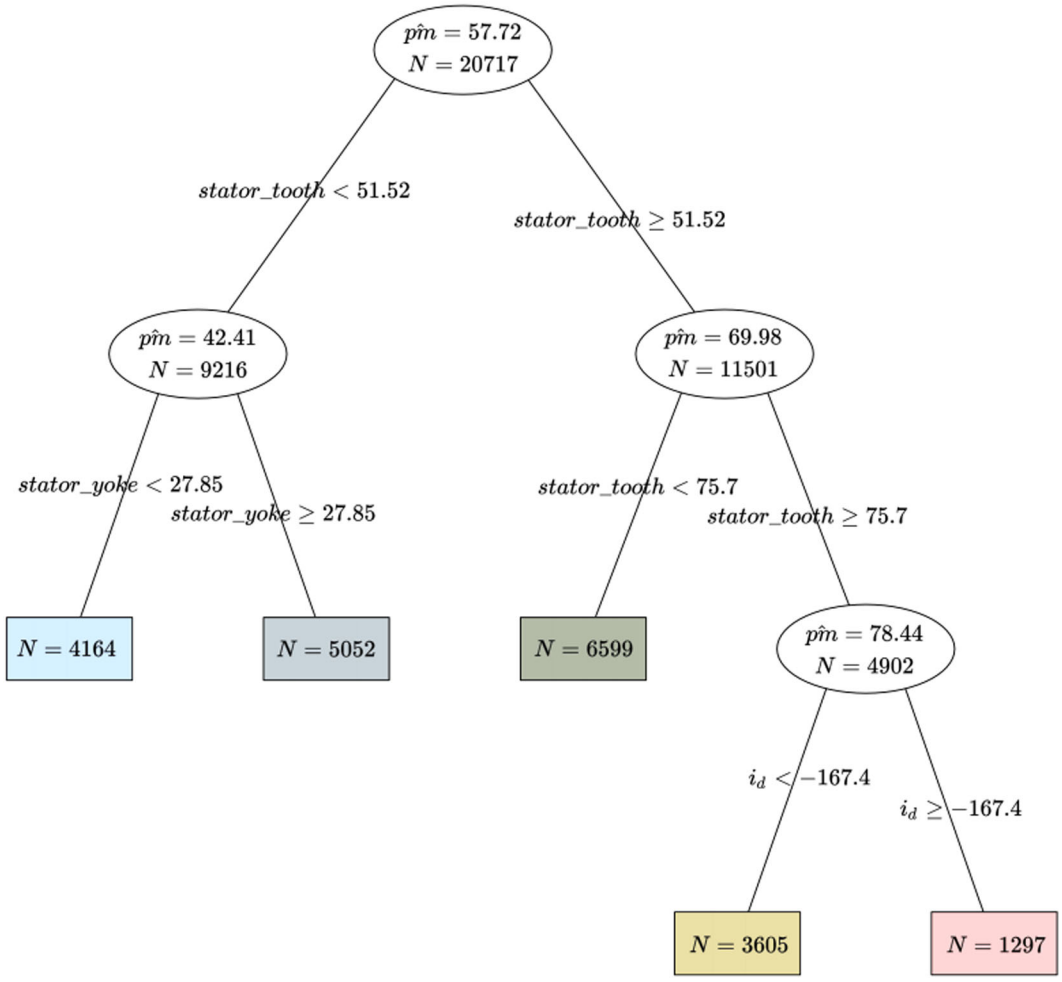

FIGURE 9 Example of the tree structure of the nonhomogeneous univariate mtDBN model in the case of the electric motor. In this case, the initial cuts are made around the yoke and tooth temperatures inside the motor. It makes sense that temperatures at other points of the motor are useful in determining our objective temperature *pm*, and for very high temperatures, the electric current i_d defines the two most extreme groups. mtDBN, model tree dynamic Bayesian network. [Color figure can be viewed at wileyonlinelibrary.com]

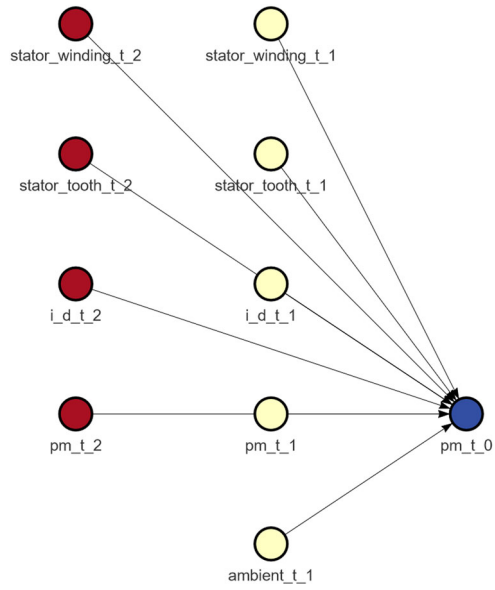

FIGURE 10 Example of the objective variable *pm* (in blue) and its parent nodes inside the DBN structure in the nonhomogeneous mtDBN model of Markovian order 2. Apart from the autoregressive component of *pm* on itself, we can see that two variables are also used in the tree structure, *stator tooth* and i_d, and two other temperatures, *stator winding* and *ambient*, are used by the DBN model to predict *pm*. Intra-slice arcs are not permitted due to the structure learning algorithm used. mtDBN, model tree dynamic Bayesian network. [Color figure can be viewed at wileyonlinelibrary.com]

determine whether we are in the case of the highest temperatures of the data set. In the case of temperatures, the relationships between the variables do not increase linearly as the temperature increases. The baseline DBN model approximates the global behaviour of the system, while the mtDBN model defines different splits based on these temperatures. The hybrid structure allows the model to switch between the leaf node models depending on the current temperatures of the system, both at the beginning of the prediction and while performing forecasting. This helps improving the accuracy of the mtDBN models.

We can also look at the DBN structure shown in Figure 10 for further insight. Given that the model is nonhomogeneous, this structure corresponds to the third leaf node, the one with the higher number of instances in the training data set. We can see that, apart from the variables used in the tree, the DBN also uses the *ambient* and the *stator_winding* temperatures to perform inference. This information could help an expert in the field understand the relationships of the variables in the problem and identify the most important physical relationships inside the system. In our case, we cannot extract a level of information as deep as in the synthetic case, but it can help us understand which are the most important variables in the system model and subsequently which interventions will affect it the most and what variables the model bases its forecasts on.

The LSTM model results remain in a similar range and are specially good in terms of training and execution time. The performance in terms of the MAE does not differ much from the rest of the models, given that this kind of model excels when trained with a large number of instances and the dimensionality is drastically reduced in the preprocessing. In this scenario, the biggest drawback we could find is in terms of interpretability of the model. The forecasts and times of the LSTM model are very competitive, but it offers no insight into how the system works and what the model bases

its predictions on. The HFCM model obtains much better results in this experiment compared to the long-term scenario of the synthetic data. The MAE and MAPE results obtained are comparable to the LSTM results but slightly worse, because it still suffers from the midterm predictions. On the other hand, the execution time is the best of all the models once again, but the training time is the slowest, even slower than the homogeneous mtDBN models.

4.3 | Real data application: Taiwan Stock Exchange Corporation (TSEC) stock index

One of the most popular real world applications of time series forecasting models is in financial data from the stock market.[27] In this experiment, we have chosen the TSEC weighted index, which aggregates the stock values of almost 900 Taiwanese companies, as our objective index. The original data was obtained from yahoo! finance (https://finance.yahoo.com/quote/%5ETWII?p=%5ETWII).

We extracted 16 months worth of daily values from the 1st of January 2021 to the 29th of April 2022. The data set is composed of five variables: the opening value of the index (*Open*), the closing value (*Close*), the maximum value during that day (*High*), the minimum value (*Low*), and the volume of transactions (*Volume*). In contrast with the other experiments, we have a reduced data set of 319 instances in total, given that the stock market closes during the weekends and on specific holidays, with four out of five variables which are deeply correlated due to being specific values that the index took during a single day. This correlation can be seen in Figure 11. In this scenario, our objective will be to forecast the opening value that the index will take on the next day. This case offers a clear contrast with the previous experiments and can be used to evaluate the performance of the mtDBN model with reduced training data and very short-term predictions. Our objective variable has a mean value of 17205, with a maximum value of 18,620 and a minimum of 14,937. We also set a minimum of 50 instances per leaf node as a safety measure, given that the number of instances in the data set is very low.

FIGURE 11 Heatmap showing the correlation between the variables in the TSEC stock data set. Most of the variables have a correlation very close to 1, except for the volume of daily transactions. TSEC, Taiwan Stock Exchange Corporation. [Color figure can be viewed at wileyonlinelibrary.com]

The results of the experiment in Table 7 show that all the hybrid mtDBN models obtain better accuracy results than the baseline DBN model in both MAE and MAPE. The MAPE results show that this is the most accurate scenario for all the fitted models, largely due to the fact that we are performing single-step forecasting. In this case, given that we are predicting only the next instant, it is not possible to perform piecewise regression, because only a single prediction with a single model is performed. As a result, the only difference between the baseline and the mtDBN models is in which DBN model is used to perform forecasting. This is clearly evidenced in Table 8, where only the homogeneous univariate mtDBN obtains statistically significant results when compared with the baseline. A univariate tree is the most effective in this experiment due to most of the variables being deeply correlated with one another. There is little improvement from adding another variable as the possible objective of the tree when they all encode redundant information. Ultimately, the univariate homogeneous mtDBN model is able to obtain statistically significantly better results than the baseline because the tree structure is able to differentiate appropriate scenarios where a specifically trained DBN model can perform a single prediction more accurately than a global DBN model.

The tree structure of the model can be seen in Figure 12. We can appreciate that the tree structure separates the values of the variable *High* into three groups: one for the instances under the mean value of the stock index, one for the instances around that value and one

TABLE 7 Results in terms of the MAE, MAPE, training time and execution time of the models for the stock index data set.

	Homogen?	Tree	MAE	MAPE	Train (m)	Exec (s)
DBN	–	–	64.55	0.385	21.08	0.009
mtDBN	Yes	Univ	55.18	0.327	20.68	0.011
mtDBN	Yes	Multiv	61.12	0.362	20.88	0.008
mtDBN	No	Univ	56.88	0.336	11.73	0.011
mtDBN	No	Multiv	61.47	0.364	13.69	0.008
LSTM	–	–	77.48	0.460	0.96	0.112
HFCM	–	–	84.87	0.504	6.58	3e−5

Abbreviations: DBN, dynamic Bayesian network; HFCM, high-order fuzzy cognitive maps; LSTM, long short-term memory; MAE, mean absolute error; MAPE, mean absolute percentage error; mtDBN, model tree dynamic Bayesian network.

TABLE 8 Resulting *p*-value of the Wilcoxon rank-sum tests for the stock index data set with respect to the regular DBN model.

Homogen?	Tree	*p*-value
Yes	Univariate	0.033
Yes	Multivariate	0.542
No	Univariate	0.193
No	Multivariate	0.120

Abbreviation: DBN, dynamic Bayesian network.

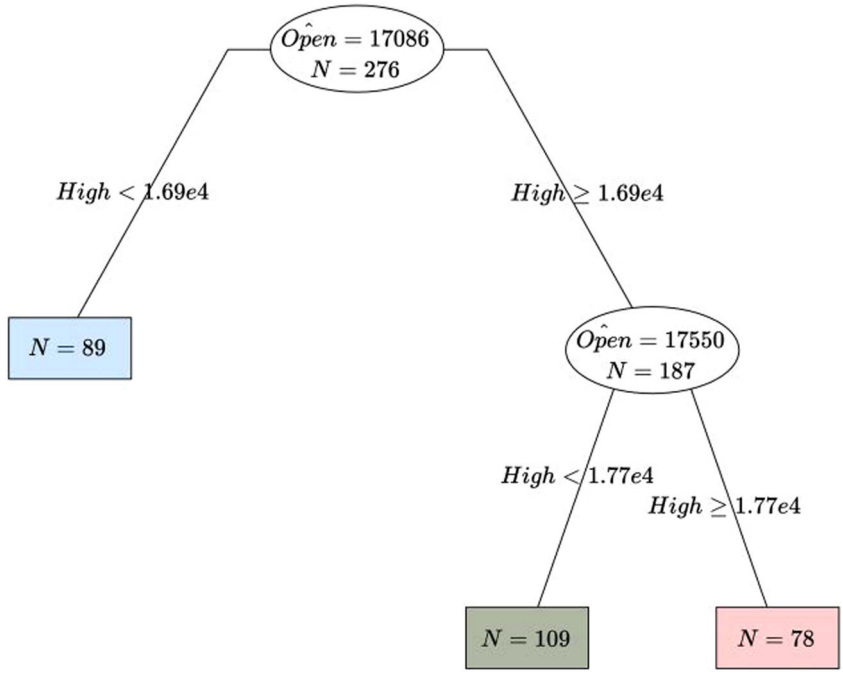

FIGURE 12 Tree structure of the homogeneous univariate mtDBN model. The splits define three cases depending on the highest value that the stock index took that day. From left to right, the leaf nodes define the cases where the stock index was lower than the mean, the cases where it was around the mean and the cases where it was higher than the mean. mtDBN, model tree dynamic Bayesian network. [Color figure can be viewed at wileyonlinelibrary.com]

for the instances above. With the information at hand, this kind of division is the most interesting one, given that stock indexes values behave differently while on their highest and on their lowest based on the effects of market operations of selling and buying. Another interesting result from the tree structure is that the number of instances in each leaf node is so low that it takes less training time to learn three networks in the nonhomogeneous models than a single network with all the instances in the homogeneous ones in Table 7. This result is not seen in the other experiments because they both have thousands of instances per leaf node, which ramps up the structure learning time.

The DBN structure in Figure 13 shows that the value of the *Open* variable is mainly decided by the last two opening values of the stock index and the closing value of the last day. It makes sense that the opening value of the next day is directly influenced by the closing value of the previous day, and using the lowest values of the index in the two previous days can help mitigating the cases where the index tanked its value one day, but it recovered on the other one.

In this case, both the LSTM and HFCM models obtain comparable accuracies to the DBN-based models. The LSTM model has the best training time, while the HFCM has the fastest execution time. The tests in Table 9 show that the results of the LSTM and the HFCM are equivalent in terms of accuracy, given that their differences are not statistically significant. When comparing all the algorithms with the Kruskal–Wallis test, we obtained a p value of 3.12e−05. Ultimately, the results obtained by the homogeneous

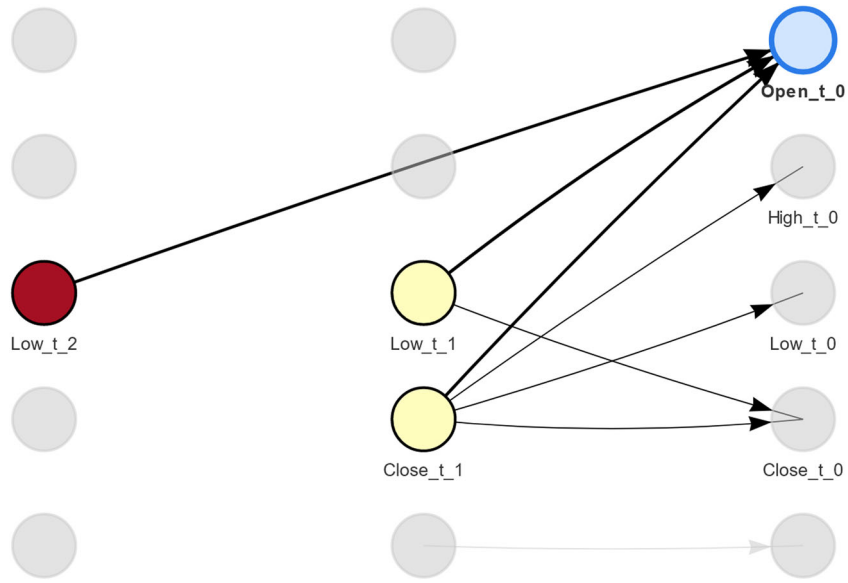

FIGURE 13 Example of the DBN structure of the homogeneous univariate mtDBN model in the stock data set. The opening index value of the next day is obtained taking into account the lowest values of the last two days and the closing value of the previous day. DBN, dynamic Bayesian network; mtDBN, model tree dynamic Bayesian network. [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 9 Results of the Wilcoxon rank-sum tests in the multiple comparisons for the stock index experiment.

	DBN	mtDBN	LSTM	HFCM
DBN	–	0.033	1.165e−3	3.280e−3
mtDBN	0.033	–	1.430e−4	4.636e−5
LSTM	1.165e−3	1.430e−4	–	0.405
HFCM	3.280e−3	4.636e−5	0.405	–

Note: The pairwise tests show that the LSTM and HFCM models do not present significant differences in their results.

univariate mtDBN are statistically significantly better than the other models for this specific scenario.

5 | CONCLUSIONS

In this article, we propose a hybrid model, mtDBN, that performs piecewise forecasting of nonlinear multivariate time series combining a model regression tree and DBN models. The tree model divides the contexts represented in the training data set, and different DBN models are fitted to each. Then, the tree structure is used as a model selector in a multinet architecture for deciding which DBN model to use in each instant of the forecast. Thus, we overcome one of the shortcomings of Gaussian DBN models when applied to real data, that they can only model

linear processes. Our results on both synthetic and real data show that the hybrid mtDBN model effectively reduces the error of a baseline DBN when applied to nonlinear problems at the cost of a higher training time and a similar execution time. The hybrid model proved to be a viable alternative when applying DBN models in this kind of setting. To make the experiments replicable, we made all code available online, and we created a simulation for the first experiment that is publicly available and used public data from real-world problems for the second and third experiments. We also offer the hybrid mtDBN model on a public repository inside an R package that makes it ready to use and deploy.

In this study, the most elemental version of the mtDBN is presented. In future work, we would like to try different splitting criteria for growing the tree model. In particular, given that we are working with time series data, it could be interesting to define a tree model that separates time series based on their similarity with a metric such as dynamic time warping.[28] Another open option is the use of a different partition method, for example, clustering or more complex tree structures such as random forests, to perform the initial data set division and classification of new instances during forecasting.

ACKNOWLEDGEMENTS

This study was partially supported by the Spanish Ministry of Science, Innovation and Universities through the PID2019-109247GB-I00 project and by the BBVA Foundation (2019 Call) through the "Score-based nonstationary temporal Bayesian networks. Applications in climate and neuroscience" (BAYES-CLIMA-NEURO) project.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in GitHub at https://github.com/dkesada/mtDBN and in Kaggle at https://www.kaggle.com/wkirgsn/electric-motor-temperature.

ORCID

David Quesada http://orcid.org/0000-0002-7280-904X
Concha Bielza http://orcid.org/0000-0001-7109-2668
Pedro Larrañaga https://orcid.org/0000-0003-0652-9872

REFERENCES

1. Aminikhanghahi S, Cook DJ. A survey of methods for time series change point detection. *Knowl Inf Syst*. 2017;51:339-367.
2. Murphy K. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis. UC Berkeley, Computer Science Division; 2002.
3. Duan T. Auto regressive dynamic Bayesian network and its application in stock market inference. IFIP International Conference on Artificial Intelligence Applications and Innovations. Springer;2016: 419-428.
4. Trifonova N, Karnauskas M, Kelble C. Predicting ecosystem components in the Gulf of Mexico and their responses to climate variability with a dynamic Bayesian network model. *PLOS ONE*. 2019;14:e0209257.
5. Liang H, Ganeshbabu U, Thorne T. A dynamic Bayesian network approach for analysing topic-sentiment evolution. *IEEE Access*. 2020;8:54164-54174.

6. Zhu J, Zhang W, Li X. Fatigue damage assessment of orthotropic steel deck using dynamic Bayesian networks. *Int J Fatigue*. 2019;118:44-53.
7. Cai B, Shao X, Liu Y, et al. Remaining useful life estimation of structure systems under the influence of multiple causes: subsea pipelines as a case study. *IEEE Trans Ind Electron*. 2019;67:5737-5747.
8. Li C, Mahadevan S, Ling Y, Choze S, Wang L. Dynamic Bayesian network for aircraft wing health monitoring digital twin. *Am Inst Aeronaut Astronaut J*. 2017;55:930-941.
9. Vaddi PK, Pietrykowski MC, Kar D, et al. Dynamic Bayesian networks based abnormal event classifier for nuclear power plants in case of cyber security threats. *Prog Nucl Energy*. 2020;128:103479.
10. Koller D, Friedman N. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press; 2009.
11. Gama J, Žliobaitė I, Bifet A, Pechenizkiy M, Bouchachia A. A survey on concept drift adaptation. *ACM Comput Surv*. 2014;46:1-37.
12. Breiman L, Friedman JH, Olshen RA, Stone CJ. *Classification and Regression Trees*. Wadsworth and Brooks; 1984.
13. Quinlan JR-. Learning with continuous classes. 5th Australian Joint Conference on Artificial Intelligence. Vol 92. World Scientific; 1992:343-348
14. Vanli N, Kozat S. A comprehensive approach to universal nonlinear regression based on trees. *IEEE Trans Sig Process*. 2014;62:5471-5486.
15. Schmidhuber J. Deep learning in neural networks: an overview. *Neural Netw.* 2015;61:85-117.
16. Kosko B. Fuzzy cognitive maps. *Int J Man-Mach Stud*. 1986;24:65-75.
17. Van Houdt G, Mosquera C, Nápoles G. A review on the long short-term memory model. *Artif Intell Rev*. 2020;53:5929-5955.
18. Orang O, de Lima e Silva PC, Guimarães FG. Time series forecasting using fuzzy cognitive maps: a survey. *arXiv preprint arXiv:2201.02297*; 2022.
19. Bilmes JA. Dynamic Bayesian multinets. Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence; 2000:38-45.
20. Larsen DR, Speckman PL. Multivariate regression trees for analysis of abundance data. *Biometrics*. 2004;60:543-549.
21. Sharma H, Kumar S. A survey on decision tree algorithms of classification in data mining. *Int J Sci Res*. 2016;5:2094-2097.
22. Quesada D, Valverde G, Larrañaga P, Bielza C. Long-term forecasting of multivariate time series in industrial furnaces with dynamic Gaussian Bayesian networks. *Eng Appl Artif Intell*. 2021;103:104301.
23. Quesada D, Bielza C, Larrañaga P. Structure learning of high-order dynamic bayesian networks via particle swarm optimization with order invariant encoding. International Conference on Hybrid Artificial Intelligence Systems. Springer;2021:158-171.
24. Feng G, Zhang L, Yang J, Lu W. Long-term prediction of time series using fuzzy cognitive maps. *Eng Appl Artif Intell*. 2021;102:104274.
25. Kirchgässner W, Wallscheid O, Böcker J. Estimating electric motor temperatures with deep residual machine learning. *IEEE Trans Power Electron*. 2021;36:7480-7488.
26. Giorgino T. Computing and visualizing dynamic time warping alignments in R: the dtw package. *J Stat Softw*. 2009;31:1-24.
27. Singh P, Huang YP. A high-order neutrosophic-neuro-gradient descent algorithm-based expert system for time series forecasting. *Int J Fuzzy Syst*. 2019;21:2245-2257.
28. Berndt DJ, Clifford J. Using dynamic time warping to find patterns in time series. Workshop on Knowledge Discovery in Databases. Vol 10. 1994; 359-370.

How to cite this article: Quesada D, Bielza C, Fontán P, Larrañaga P. Piecewise forecasting of nonlinear time series with model tree dynamic Bayesian networks. *Int J Intell Syst*. 2022;37:9108-9137. doi:10.1002/int.22982

APPENDIX A: ORDINARY DIFFERENTIAL EQUATION SYSTEM FOR THE FOULING PHENOMENON

The main idea of our simulation is to generate an environment where we have a fluid flowing through a tube and a heat source heating the walls of said tube. Meanwhile, as this fluid is heated, it precipitates materials that create an insulating layer inside the tube over time. This is a process called fouling.[22]

Our aim is to define a simplified version of this phenomenon, one that reflects the nonlinear relationships of the variables and respects the underlying physical process. Our first simplification is that the simulation does not have a spatial component, so heat will transfer from the heat source to the tube walls by radiation from a single point to another and from the tube walls to the fluid by convection. This generates a heating component that increases the temperature of the fluid. As the fouling process occurs, the thermal conductivity of the system decreases, and the heat transferred from the tube walls to the fluid is reduced. To dissipate heat from the system, we have a flow component that renews fresh cooler fluid at each instant. All fluid volumes are heated as a whole and move in and out of the system as a singular unit. Initially, the fluid heats easily, but as the fouling layer grows, it becomes increasingly harder to keep the fluid temperature high.

The first component that we define is the growth of the insulating layer over time:

$$\frac{\partial S_c}{\partial t} = A_1 k_1 C_a, \tag{A1}$$

where S_c is the thickness of the insulating layer, t is the time, $A_1 > 0$ is a control constant, k_1 is the reaction speed at which particles prone to fouling precipitate and C_a is the concentration of particles prone to fouling in the fluid. Equation (A1) controls the rate at which the fouling layer grows and is dependent on the value of C_a, which is a property of the fluid, and on k_1. The value of k_1 is defined by

$$k_1 = A_2 e^{\frac{-A_3}{RT_1}}, \tag{A2}$$

where $A_2 > 0$ is a constant pre-exponential factor, $A_3 > 0$ is an activation energy constant, R is the ideal gas constant and T_1 is the temperature of the fluid. A higher fluid temperature will accelerate the growth of the insulating layer.

The second component is the evolution of the concentration of particles prone to fouling C_a:

$$\frac{\partial C_a}{\partial t} = -A_4 k_2 C_a, \tag{A3}$$

$$k_2 = A_2^{'} e^{\frac{-A_3^{'}}{RT_1}}. \tag{A4}$$

The process is very similar to the previous one, but in Equation (A3), the concentration diminishes each instant. In this case, a high T_1 increases the consumption rate of C_a, but it also affects the growth rate of S_c.

Once we have modelled the fouling layer and the concentration of particles that generate it, we can define the evolution of the fluid temperature T_1 and the flow component:

$$\rho_1 C_{p1}\left(\frac{\partial T_1}{\partial t} - A_5 Q_{in}\Delta T\right) = f_1(T_1, T_2), \tag{A5}$$

$$f_1(T_1, T_2) = \frac{A_6}{S_c}(T_2 - T_1), \tag{A6}$$

$$Q_{in} = vol\frac{\pi(2r)^2}{4}. \tag{A7}$$

In Equations (A5) and (A6), ρ_1 is the density of the fluid, C_{p1} is the thermal capacity of the fluid, Q_{in} represents the flow of new fluid inside the system each instant, $\Delta T = T_{in} - T_1$ is the difference of temperature between the fresh fluid entering the system and the fluid currently in the system, T_2 is the temperature of the tube wall and A_5 and A_6 are control constants. In essence, the convective component of the equations will transfer heat from the tube wall to the fluid, and as S_c from Equation (A1) increases, this convective component will degrade. The flow component Q_{in} in the system adds the effect of cold fluid entering the system each instant through the area of the tube.

In Equation (A7), *vol* is the volume of fluid going through the system each instant, and r is the radius of the tube. The effect of this component is written in Equation (A5) as the product $A_5 Q_{in}\Delta T$ of the flow rate and the difference in the fluid temperature ΔT. The cold fluid entering the system translates into a loss of temperature in the fluid from the previous instant. This is of vital importance because it adds a mechanism to reduce the temperature of the system. Without it, the simulation will only increase its temperature monotonically. Now, with both S_c and the new flow, there can be situations where the heat transfer between T_2 and T_1 is so low that the fluid loses temperature over time due to the effect of the colder flow. It also adds the volume inside the tube as a variable that can be manipulated by an agent to perform interventions in the system.

The last equations define the temperature of the tube walls T_2 and the temperature of the furnace T_3:

$$\rho_2 C_{p2}\left(\frac{\partial T_2}{\partial t}\right) = f_2(T_1, T_2, T_3), \tag{A8}$$

$$f_2(T_1, T_2, T_3) = \frac{A_6}{S_c}(T_1 - T_2) + A_7\left(T_3^4 - T_2^4\right), \tag{A9}$$

$$T_3 = T_{min} + \frac{1}{1 + e^{-m_c}}(T_{max} - T_{min}). \tag{A10}$$

In Equations (A8) and (A9), T_2 is defined by its interactions with T_1 and T_3, the latter being the main factor that defines the value of T_2 because T_3 represents the temperature inside the furnace. In the previous equations, ρ_2 is the density of the tube wall alloy material, C_{p2} is its thermal capacity and A_7 is a control constant. The tube wall temperature is raised by radiation from T_3 and is decreased by induction due to the colder fluid inside, but this effect is more insignificant than the radiation.

In Equation (A10), we define how T_3 is calculated. The furnace temperature changes inside a range defined by a minimum temperature T_{min} and a maximum T_{max} depending on

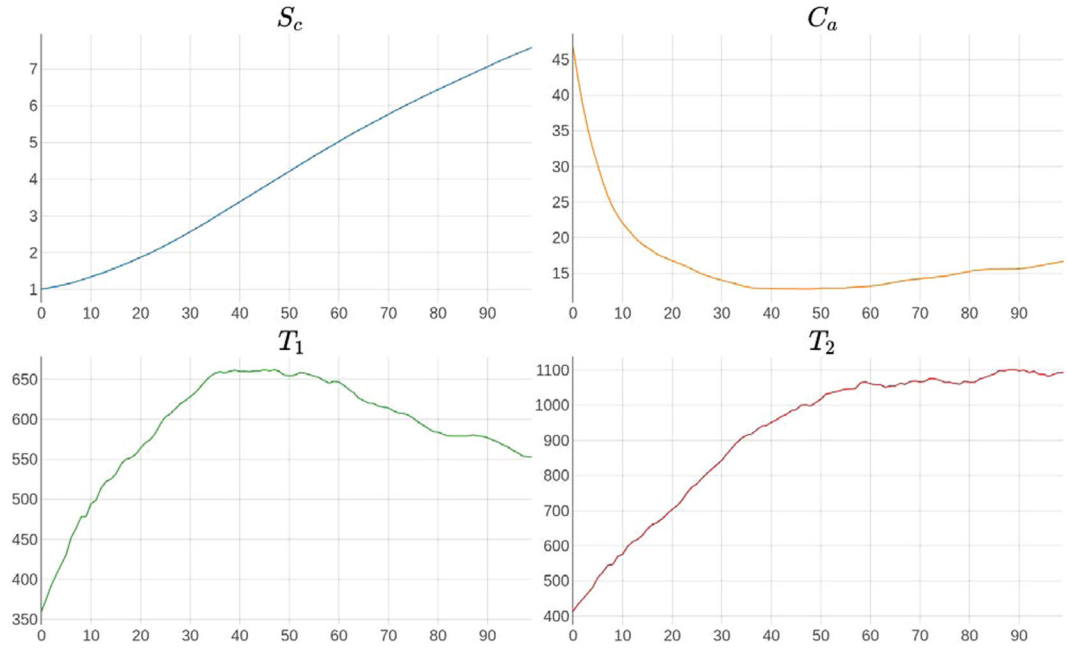

FIGURE A1 Example of a 100 time instant trace created by the simulation. Several time series with data and some noise of each variable are generated.

the amount of fuel m_c administered to the furnace heaters. This simulates the behaviour of a valve with a sigmoid function, where an operator can modify the heat inside the furnace by varying m_c.

The variables that can be manipulated to generate traces from the same process are the fluid properties such as the initial values of S_c and C_a, the values of ρ_2 and C_{p2}, the furnace state variables such as m_c and the volume *vol*. These can be modified to obtain different behaviours in the traces generated, but the other parameters remain constant. The idea is to circulate different types of fluids through the same furnace and increase or decrease the temperatures inside the furnace through interventions on m_c and *vol*.

With this ODE system, we can generate a synthetic data set with an arbitrary number of traces from the fouling process and use it to train our models. The code of the simulation and the functions to generate the seeded datasets are readily available in a public repository online (https://github.com/dkesada/mtDBN). An example of the kind of traces it generates is shown in Figure A1.