Journal of Signal Processing Systems (2019) 91:191–215
https://doi.org/10.1007/s11265-018-1376-5

Multi-Output Tree Chaining: An Interpretative Modelling and Lightweight Multi-Target Approach

Saulo Martiello Mastelini[1] · Victor Guilherme Turrisi da Costa[1] · Everton Jose Santana[2] · Felipe Kenji Nakano[3] · Rodrigo Capobianco Guido[4] · Ricardo Cerri[3] · Sylvio Barbon Jr.[1]

Received: 15 September 2017 / Revised: 16 January 2018 / Accepted: 27 April 2018 / Published online: 5 May 2018

Abstract

Multi-target regression (MTR) regards predictive problems with multiple numerical targets. To solve this, machine learning techniques can model solutions treating each target as a separated problem based only on the input features. Nonetheless, modelling inter-target correlation can improve predictive performance. When performing MTR tasks using the statistical dependencies of targets, several approaches put aside the evaluation of each pair-wise correlation between those targets, which may differ for each problem. Besides that, one of the main drawbacks of the current leading MTR method is its high memory cost. In this paper, we propose a novel MTR method called Multi-output Tree Chaining (MOTC) to overcome the mentioned disadvantages. Our method provides an interpretative internal tree-based structure which represents the relationships between targets denominated Chaining Trees (CT). Different from the current techniques, we compute the outputs dependencies, one-by-one, based on the Random Forest importance metric. Furthermore, we proposed a memory friendly approach which reduces the number of required regression models when compared to a leading method, reducing computational cost. We compared the proposed algorithm against three MTR methods (Single-target - ST; Multi-Target Regressor Stacking - MTRS; and Ensemble of Regressor Chains - ERC) on 18 benchmark datasets with two base regression algorithms (Random Forest and Support Vector Regression). The obtained results show that our method is superior to the ST approach regarding predictive performance, whereas, having no significant difference from ERC and MTRS. Moreover, the interpretative tree-based structures built by MOTC pose as great insight on the relationships among targets. Lastly, the proposed solution used significantly less memory than ERC being very similar in predictive performance.

Keywords Multi-target regression · Multi-output · Memory-friendly algorithm · Interpretative tree structure · Machine learning

1 Introduction

The concept of machine learning (ML) denotes a wide set of techniques that have been explored in multiple fields with a diverse set of objectives. One of those research fields is the signal processing, which consists of several tasks, such as discovering important features from noisy signals,

✉ Saulo Martiello Mastelini
mastelini@uel.br

Victor Guilherme Turrisi da Costa
victorturrisi@uel.br

Everton Jose Santana
santana.everton@ieee.org

Felipe Kenji Nakano
felipe.nakano@dc.ufscar.br

Rodrigo Capobianco Guido
guido@ieee.org

Ricardo Cerri
cerri@dc.ufscar.br

Sylvio Barbon Jr.
barbon@uel.br

[1] Computer Science Department, State University of Londrina. Rodovia Celso Garcia Cid, Km 380, s/n - Campus Universitário, Londrina, PR, 86057-970, Brazil

[2] Electrical Engineering Department, State University of Londrina. Rodovia Celso Garcia Cid, Km 380, s/n - Campus Universitário, Londrina, PR, 86057-970, Brazil

[3] Department of Computer Science, Federal University of São Carlos, Rodovia Washington Luís, km 235, São Carlos, SP 13565-905, Brazil

[4] Instituto de Biociências, Letras e Ciências Exatas, Unesp - Univ Estadual Paulista (São Paulo State University), Rua Cristóvão Colombo 2265, Jd Nazareth, 15054-000, São José do Rio Preto, SP, Brazil

enhancing signal quality and predicting signals properties [4, 5, 8, 10, 14, 18, 24]. One frequently performed task consists of predicting a target, response, or output, y based on a set X_m of m input variables. These target values can be categorical or numerical, resulting in classification or regression problems, respectively. Categorical targets assume a finite set of L possible output values. When $|L| = 2$, the resulting task is denominated a binary predictive problem. Whenever $|L| > 2$, the obtained problem is called a multiclass classification task. In contrast, when regression tasks are considered, the output variable assumes ordered and infinite numerical values.

When a learning task focus on predicting a unique target, we have a *single-target* (ST) problem. On the other hand, many machine learning problems consist of multiple targets, which are denominated as *multi-target* (MT) problems [2, 17]. MT includes many types of prediction tasks depending on the type of the responses, i.e., multi-label, multi-dimensional, hierarchical classification, and multi-target regression [2, 17, 21].

Consider a description space X that consists of i tuples with primitive data types (discrete or continuous values) in the form $X_i = \{x_{i_1}, x_{i_2}, ..., x_{i_m}\}$, where m represents the number of descriptive features. Also, consider a target space Y in the form $Y_i = \{y_{i_1}, y_{i_2}, ..., y_{i_d}\}$, being d the number of problem's outputs. In addition, take a set of examples E comprising elements from X and Y, i.e., $E = \{(X_i, Y_i)\}$, $1 \leq i \leq N$, where N is the number of examples in E. Lastly, take into consideration a quality criterion q which rewards prediction models with high predictive performance and low complexity [17]. A MT problem consists in finding a function $f : X \rightarrow Y$ such that f maximises q. The mentioned prediction problem is solved upon the set of know examples, E. In a MT problem, f can either refer to a single prediction model or a set thereof.

Depending on Y, different denominations are given to the modelling. When the elements in Y assume binary values, the resulting task is called *Multi-label* classification [17], in contrast to *multi-dimensional* or *multi-target* classification, which is related to the prediction of multiple discrete or categorical values that assume more than two single values [17, 21]. In *hierarchical* classification tasks, the multiple outputs are arranged into a directed acyclic graph or in a tree hierarchy [17]. When the elements of Y are continuous, with the possibility of being statistically correlated, the related predictive problem is called *multi-target regression* (MTR) [1, 2, 21, 25, 26].

In some cases, one could simply consider that the targets have no underlying relational properties and treat the prediction of each target as a separated predictive task, applying a ST technique. This comprehends to the extensive set of solutions based on machine learning, e.g., Artificial Neural Networks (ANN), Decision Trees (DT), Bayesian Models and Support Vector Machines (SVM). However, this assumption causes a negative impact on prediction performance since the inter-correlation between targets are ignored. In fact, many real-life problems such as the prediction of multiple vegetation, soil, water and river flow properties, the estimation of monthly online product sales, the price forecasting of multiple products in a supply management chain and of air-tickets for different companies, and even solar flares [2, 15, 25], are reported as MTR tasks. Whenever targets have dependencies among themselves, the easier to predict can be used as additional features for the more challenging ones. Nevertheless, the ST strategy has been employed across multiple MTR papers as a base level method for comparisons [2, 21, 25].

In the context of ML for signal processing, many works have been published in the past years [4, 5, 8, 24, 27, 28]. In these works, regression models have been broadly applied in time series, text, speech and image processing, for instance in the determination of sound sources direction-of-arrival, estimation of the head pose, improvement of low-resolution images and prediction of stock prices. In these kind of tasks, the computational costs of a solution can limit its application even if it has high predictive capabilities.

According to Borchani et al. [2], MTR tasks are solved with two general approaches: algorithm adaptation and problem transformation. Kocev et al. [17] denote the methodologies as global and local methods, respectively. The former refers to the adaptation of well known ST regression algorithms to deal with multiple outputs while additionally investigating the possible relationships among targets. In the latter, the training data is manipulated, and common regression techniques to predict ST problems are then employed. Several MTR methods belonging to both approaches were proposed in past years [1, 2, 17, 21, 22, 25]. Some of the local approaches had achieved the highest performance in multiple MT scenarios, outperforming global methods [21, 22, 25, 26]. However, the actual state-of-art methods present some limitations in the interpretation of targets dependencies and have high computational cost solutions.

In this paper, we propose a novel local method, called Multi-output Tree Chaining (MOTC), which uses a tree-based representation to model the statistical dependencies between targets. Our goal is to address the aforementioned disadvantages by creating a highly interpretable with high predictive performance method, whereas being memory and time conservative when constructing the chaining trees. MOTC differs from other local-based methods in the sense that it uses only the existing dependency relationships between outputs as additional information for regression, with a sophisticated and yet lightweight approach. Also, we propose to use the Random Forest variable importance metric to model nonlinear statistical dependencies between targets. Using the Hoeffding Bound concept [13], we select

groups of outputs which are the most correlated with a specific response, without memory-time trade-offs.

Through an extensible experimental evaluation of MOTC against three other local-based MTR methods (Single-target, Multi-target Regressor Stacking and Ensemble of Regressor Chains) in 18 benchmark datasets along with two regression techniques (Random Forest and Support Vector Machine), we demonstrate that our method can achieve high prediction performance, whereas generating a compact number of regression models when compared with ERC, reducing the memory cost. Besides, we present a combined representation of the generated MOTC's chaining trees to give some insights about how they could be employed as an interpretation tool. In this way, it is possible to discover linear and nonlinear target relations and contribute toward further advances in signal processing technologies.

The rest of this paper is structured as follows. Section 2 reviews related works in MTR. In Section 3, our method (MOTC) is presented. Section 4 presents the materials and the experimental setup employed in our experiments. In Section 5 we discuss the obtained results, comparing MOTC with other MTR approaches. We present the final considerations in Section 6. Lastly, two Appendices with the description of the datasets used in the work and the condensed representations of the chaining trees obtained during the experiments are included.

2 Related Works

A first directly derived approach of problem transformation is to predict each target variable separately, as multiple independent ST problems. Although pretty straightforward, this method outperforms MTR methods (both those based on algorithm adaptation and problem transformation) in many cases [2, 25]. Nonetheless, ST approaches do not explore the assumed target correlations, so using an MTR strategy ought to achieve better results when there are target co-dependencies. Although more than one predictor is used to represent an MTR problem, which leads to negatively impacting the generated model interpretation and increasing computation training costs, this kind of approach offers multiple advantages. Firstly, the possibility of using any base regressor, or even a hybrid set of regressors, could lead to better prediction performance and exploration of a particular problem characteristics. Besides that, problem adaptation methods improve the resulting model modularity and conceptual simplicity, presenting better accuracy than state-of-the-art methods [21, 22, 25].

In past years, some problem transformation methods were proposed to exploit inter-target properties as well. Zhang et al. [29] proposed the modification of the input space of a task through a visualisation technique so that a MTR task could be modelled as a wider ST problem. The authors used a Support Vector Regression (SVR) machine and achieved results comparable to the ST strategy. Tsoumakas et al. [26] created random linear combinations of the targets to explore the relation between output values. This approach increases the original feature space dimension and solves multiple ST problems in the transformed space. After that, the values predicted are used to solve a linear system to obtain the original target predictions. Their results compared favourably both for the ST approach and the global-based solution Multi-objective Random Forest (MORF), firstly described in Kocev et al. [16].

Some MTR methods were also adapted from multi-label classification (MLC) [2, 25]. Spyromitros-Xioufis et al. [25] proposed two main methods inspired by MLC: Multi-Target Regressor Stacking (MTRS) and Ensemble of Regressor Chains (ERC). Both have the core concept of applying targets approximations as additional, or expanded, features, as proposed in the Cascade Generalisation [11]. These methods are better described in Sections 2.1 and 2.2.

A few recent works were inspired by the idea of the MTRS, with the premise that deeper layers of stacked ST predictors could offer better predictive performance than using just one layer (ST) or two layers (MTRS). Santana et al. [23] proposed the Deep Regressor Stack (DRS), which uses the targets estimations as additional predicting features in a naive deep learning method. In DRS, the predictions of the targets that obtained less error on a validation dataset are sequentially added as new features, with a fixed number of layers. Their approach presented some gains over the other compared MTR solutions, though this method brought an overhead both in time and memory.

Another promising novel approach, called Deep Structure for Tracking Asynchronous Regressor Stacking (DSTARS) was proposed by Mastelini et al. [20]. By defining the best number of regression layers in an adaptive manner, DSTARS is able to model different levels of interaction between targets and could bring some clues about their relationships. DSTARS has the drawback of increasing computation memory and time cost due to the use of deeper layers of stacked ST predictors, as well as, a processing stage to find the adequate adequate number of layers built for each output.

Aiming at improving the ERC algorithm, two local-based MTR methods were proposed recently. Moyano et al. [22] proposed the use of Genetic Algorithm (GA) to find an improved set of target chains and to explore the linear correlation among target variables. Their method performs better than ERC in some scenarios, but solving the combinatory problem brings a new processing overhead to the already very costly ERC algorithm.

Melki et al. [21] proposed the Maximum Correlated Chain method with a single group of chained ST regressors

instead of an ensemble of chain predictors. They computed the linear correlation between each pair of target variables to determine the order in which models would be induced. Each model, in the same way as other local-based approaches, uses the preceding regressors predictions as additional features. The authors report good prediction results when comparing their approach with ERC, while also converting the set of chains to a single group, reducing computational time and memory complexities.

Both ERC-based approaches use the linear correlation to explain the underlying relationships among target variables. Despite the assumption of linear dependency is enough in many problems, some tasks are subject to present non-linear statistical relationships between targets. In those tasks, non-linear behaviour mapping should reflect in better problem comprehension and greater predictive performance. Besides that, the aforementioned works assume that the maximisation of the summed correlation within chains is a complete descriptor of the inter-responses dependency behaviour. They do not consider different degrees of correlation between them, as each target is preceded only by a unique assumed less correlated response, in a queue-like fashion.

Our approach overcomes the mentioned gaps in ERC-based approaches. By employing the Random Forest (RF) importance metric, MOTC is able to quantify non-linear relations among target variables. Also, by representing the dependencies among outputs in a tree based representation, multiple levels of dependency are explored. Our tree chaining idea combines both ERC's chains with MTRS's stacked predictors. A path from root node to a leaf could be interpreted as a regression chain, where each level in the chaining tree is equivalent to a MTRS model. MOTC is still able to reduce computational complexities by employing the statistical measure called Hoeffding bound over the measured inter-target dependencies. With this procedure, only the most significant outputs are selected to recursively compose the chaining tree. Lastly, our approach results in highly interpretative models, which enables the study of how targets influence each other in problem-specific domains.

Before presenting the MOTC algorithm, the MTRS and ERC are explained in detail in the following sections, since they provided the conceptual basis to many problem transformation methods, ours included. Besides that, both algorithms are also used in our experimental comparisons against MOTC.

2.1 Multi-target Regressor Stacking

MTRS approach concerns additional input features from ST models predictions. In this sense, considering a dataset composed by $X = \{x_1, x_2, ..., x_m\}$ input features and $Y = \{y_1, y_2, ..., y_d\}$ target variables, MTRS adds the $Y' = \{y'_1, y'_2, ..., y'_m\}$ ST predictions as new inputs, creating an augmented training dataset $X' = \{x_1, x_2, ..., x_m, y'_1, y'_2, ..., y'_d\}$. The new dataset is used by each target to train another ST predictors layer, whose outputs are the final predictions of MTRS.

New instances are initially subjected to the first layer of predictors to obtain the output approximations. These outcomes are then merged with the original features which are applied towards the second level of predictors. Therefore, MTRS introduces inter-target relationships by stacking ST models to enhance tasks description and increase prediction performance. However, MTRS does not foresee the modelling of different levels of statistical relationships between outputs, neither does it considers the possible absence of dependencies between them. Also, models generated by the MTRS do not provide insightful information about inter-target properties. The MTRS training procedure is presented in Algorithm 1, where X and Y represent the input and output variables and d the number of targets.

Algorithm 1 MTRS training algorithm

```
function MTRS(X, Y, d)
    // To store the ST predictions
    Y' ← {}
    // First layer of ST models
    Level_0 ← {}
    // Regression model inducing for each target
    for t = 1 to d do
        // ST model induction
        h : X → Y^t
        Y'^,t ← predict(h, X)
        Level_0 ← {Level_0, h}
    end for
    // Augmented training set definition
    X' ← X||Y'
    // Second layer of ST models
    Level_1 ← {}
    for t = 1 to d do
        h : X' → Y^t
        Level_1 ← {Level_1, h}
    end for
    mtrs ← {Level_0, Level_1}
return mtrs
end function
```

2.2 Ensemble of Regressor Chains

The idea behind ERC is to build a set of randomly generated chained ST regressors for each target. Initially, for each chain, a ST model is induced using the first output of the sequence. New models are then induced by

following the chain order. Each new regressor is trained over the augmented input dataset composed by original input features and the predictions from the preceding regressors in the chain. After training all models, the predicted value of a new instance is the average value obtained from the chain regressors. In other words, the prediction of ERC, for a target y_i, $i = [1, d]$, is the average of the y_i predicted values over all chains.

Since the output predictions are composed of values from different sorted chains, multiple levels of combinations and inter-dependence between targets are explored. ERC creates all possible target permutations if their number is less than ten ($d \leq 3$), otherwise, the authors suggest using the fixed value of ten random combinations. The random nature of chain creation does not take into consideration possible relations between targets if they exist. In addition, there is no clear way of interpreting the interactions of responses expressed by a ERC trained model.

The training step of ERC is presented in Algorithm 2. The *permute* procedure is a function that receives a set of elements and returns all possible permutations among them. It is possible to perform all possible permutations without a limiting parameter, or to specify the maximum number of combinations as an argument (in the original formulation of ERC, the maximum is ten). In the algorithm, X and Y represent the input and output variables and d is the number of targets.

Algorithm 2 ERC training algorithm

```
function ERC(X, Y, d)
    targets ← names(Y)
    if d ≤ 3 then
        Chains ← permute(targets)
    else
        Chains ← permute(targets, 10)
    end if
    for chain in Chains do
        models_chain ← {}
        // To build augmented training sets
        X_aug ← X
        for t in chain do
            // ST model induction
            h : X_aug → Y^t
            y_pred ← predict(h, X_aug)
            // Extend training set with ST predictions
            X_aug ← X_aug || y_pred
            models_chain ← {models_chain, h}
        end for
        erc ← {erc, models_chain}
    end for
return erc
end function
```

3 Multi-output Tree Chaining

As discussed previously, ERC explores different combinations, and thus, various hypotheses of inter-dependencies between targets. This is achieved by randomly ordering the outputs and building a set of chained models. This strategy, besides being time and memory costly, does not sufficiently explore the target combinations and is even subjected to assume non-existent relationships among the outcomes. The method proposed here (MOTC) was projected aiming at both reducing memory consumption and improving the target dependency representation.

In order to truly model different levels of statistical dependencies between target variables, we propose the use of structures in the form of trees to guide the regression models induction. The resulting structures, called Chaining Trees (CTs), are constructed by sequentially evaluating which targets are more correlated with the response being evaluated using the RF_{imp}. MOTC is divided into two steps: a **top-down** CT construction and a **bottom-up** regression model induction. In the top-down step, for each target, a CT is built by considering how the outputs are related to each other. Each node in a CT corresponds to a regression model, which is trained during the bottom-up step.

Assume an MTR problem with X input variables and four targets, namely **A**, **B**, **C** and **D**. Figure 1 shows a CT for this example when considering **A** as the desired output. It is possible to see **A**, at Level 0, as the root node and also the targeted variable. Targets **B** and **C**, in this example, represent the outputs which are most correlated to **A**. Additionally, **B** can be better explained using **A** and **D**. This process continues until a maximum CT's depth state is reached, in this case, a maximum level equal to two. It is important to highlight that **A** appears both as leaf and root node, stopping the iteration to avoid unnecessary modelling, once **A** had already been treated. Chaining trees are constructed in a top-down approach, i.e., from root to leaves, as indicated in Fig. 1.

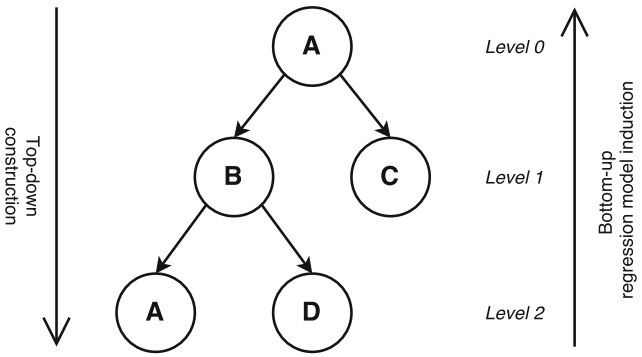


Figure 1 An hypothetical simple example of CT.

After creating the CT, MOTC performs its final modelling. It consists of inducing ST models in a bottom-up fashion, i.e., from leaves to the root node, as presented in Fig. 1. The leaf nodes in the CT correspond to standard ST models, which use only X to describe the desired output. In this sense, ST models are induced for all leaf nodes (**A**, at Level 2, **D** and **C**). After that, MOTC induces a model for **B** using an augmented training set which is created by merging X with the predictions of **A** and **D** done by the previously trained models. Then, at Level 0, **A** is induced by using as input X and the approximations of **B** and **C** (both at Level 1).

It is clear that a metric should be employed when describing dependencies between target variables and constructing the CTs. Some recent works used the linear correlation coefficient to explain target dependencies [21, 22]. Other measures could be employed for the same purpose, leading to different representations and predictive performances. In our experiments, we chose to use the Random Forest variable importance (RF_{imp}). This choice was based on the fact that this metric was reported as a reliable feature selector, which explores the impact of each variable on the whole Random Forest ensemble performance [12]. Furthermore, RF_{imp} does not suppose a linear behaviour between inputs and outputs of a predictive task, making it nonlinear. It also reports variables with no contribution or that disrupted a target estimation within a regression model. These variables are not considered then building CTs. For more details about the RF_{imp} calculation, please refer to Section 4.2.1.

In many cases, all outcomes can offer some degree of contribution to explaining other targets. Considering that MOTC aims at reducing the number of trained regression models, it must only select the most relevant outputs during the construction of the CT. For this task, we propose the use of the Hoeffding Bound (HB) to define an interval where the evaluated target dependencies are more relevant.

The Hoeffding Bound theorem [13] states that a variable v whose range is R, which was independently observed n times having a mean $\overline{v}$, has a true mean of at least $\overline{v} - \epsilon$ with statistical probability $1 - \delta$ (the error probability stated), where:

$$\epsilon = \sqrt{\frac{R^2 \, ln(\frac{1}{\delta})}{2n}}. \tag{1}$$

To limit the ramification factor of each split, only targets whose importance fluctuate around the true importance of the best feature are selected. In this sense, assuming that the observed importance of the best feature is $\overline{imp_{\text{bt}}}$, we only select targets with importance greater than $\overline{imp_{\text{bt}}} - \epsilon$.

Algorithm 3 presents the general MOTC approach. We defined the default maximum depth for CTs ($\text{Max}_{\text{depth}}$) as $2 \times log_2(d)$ if $d < 6$, and $log_2(d)$ otherwise, being d the number of target variables. Empirically, we found these values as a compromise between predictive performance and complexity reduction. In the algorithm, X and Y represent the input and output variables respectively; hb is the calculated Hoeffding Bound, and $\text{Max}_{\text{depth}}$ is the parameter for controlling the maximum depth of the CT generated. Lastly, the procedure *importance* calculates the statistical dependencies between targets, in our case, it uses the RF_{imp} measure.

Algorithm 3 MOTC algorithm

```
1:  function MOTC(X, Y^{1..d}, hb, Max_depth)
2:      motc ← {}
3:      # Influence assessment of each target on the others
4:      Imp ← importance(Y)
5:      for target in names(Y) do
6:          # Top-down step
7:          chain_tree ← GetChainTree(target, hb, Imp, Max_depth)
8:          # Bottom-up step
9:          chain_models ← BuildChainTree(X, Y, chain_tree)
10:         motc ← {motc, (chain_tree, chain_models)}
11:     end for
12: return motc
13: end function
```

The top-down building procedure of a CT, presented in Algorithm 4, can also be set to ignore the HB when defining the tree structure. This fact is treated in line 5 of Algorithm 4. If one configures MOTC to perform this, branch factor will be greater since there is no target filtering. The resulting CTs will be larger, negatively impacting in the training time and memory costs. This was not done in our experiments, since computational complexity reduction was one of our goals. Although this evaluation is out of the scope of this work, it is possible to evaluate the use of a larger model to increase the predictive performance of a given MTR problem. The presented procedure starts with a tree with a single node, the root one, which corresponds to the desired target variable. The top-down step builds a tree structure hierarchy by adding the most relevant outputs for a target as its descendents, according to a relevance interval limited by the HB. The recursive callings of this procedure end when the maximum defined depth is reached. Algorithm 4

receives the desired target for the CT building, hb which corresponds to the Hoeffding Bound, Imp which is related to the outputs importance values, and the maximum depth for the CT, $\text{Max}_{\text{depth}}$.

Algorithm 4 MOTC's top-down step

```
1: function GETCHAINTREE(target, hb, Imp, Max_depth)
2:     function CTREE(ctree, t, level)
3:         # The tree construction did not achieved the maximum defined depth
4:         if level < Max_depth then
5:             # Filtering the most relevant targets with Hoeffding bound
6:             # If no Hoeffding bound is passed all target with positive importance are used
7:             if hb ≠ null then
8:                 relevant ← which(Imp^t ≥ max(Imp^t) − hb)
9:             else
10:                relevant ← which(Imp^t > 0)
11:            end if
12:            # Recursively explores each relevant target
13:            for r in relevant do
14:                # Add r as t's descendent
15:                ctree.addChild(t, r)
16:                # Recursively calls the tree build method for the relevant output variables
17:                ctree ← CTree(ctree, r, level + 1)
18:            end for
19:        else
20:            # Leaf node: max size reached
21:            ctree.addChild(t, {})
22:        end if
23:        return ctree
24:    end function
25:    # Recursively builds chaining tree
26:    tree ← CTree(Tree(root ← target), target, 1)
27:    return tree
28: end function
```

Lastly, we present the pseudo-code for the bottom-up procedure done by MOTC in Algorithm 5. This step recursively travels a CT until it reaches the deepest nodes. After that, regression models for the leaves are constructed with only the original attributes. Then, the same operation is done while moving up in the tree. The inner nodes, including the tree root, depend on the predictions of the lower models, i.e, each model stacks the predictions of its descendents as new input features. In this sense, the bottom-up procedure assigns a regressor for each node in the CT. The MOTC's bottom-up procedure receives as parameters X and Y, which correspond to the input and output variables, respectively, and the previous built CT, $chain_{tree}$.

Algorithm 5 MOTC's bottom-up step

```
1: function BUILDCHAINTREE(X, Y, chain_tree)
2:     # Hash table to save trained ST models
3:     ChainHash ← HashTable()
4:     procedure BCTREE(node)
5:         if isLeaf(node) then
6:             # ST model induction
7:             h : X → Y^node.target
8:             ChainHash[node] ← h
9:         else
10:            desc_models ← {}
11:            for child in getDescendents(node) do
12:                BCTree(child)
13:                desc_models ← {desc_models, ChainHash[child]}
14:            end for
15:            X_aug ← X
16:            for model in desc_models do
17:                # Augments training set with the most relevant targets predictions
18:                X_aug ← X_aug || predict(model, X)
19:            end for
20:            # Induce non-leaf node model with augmented training set
21:            h : X_aug → Y^node.target
22:            ChainHash[node] ← h
23:        end if
24:    end procedure
25:    # Bottom up chaining tree model induction
26:    BCTree(root(chain_tree))
27:    return ChainHash
28: end function
```

It is worth highlighting that since all leaves in a target CT correspond to ST models without augmented training sets, these regressors are shared between all CTs. In this sense, we avoid redundant training of models, which speeds up computation while also preventing memory waste.

4 Materials and Method

This section describes the experimental setup of this work. Initially, it presents the datasets explored. It also exposes the regression techniques and the motivation of their choice. Lastly, the metrics to evaluate the results are presented.

4.1 Datasets

To evaluate the proposed algorithm, 18 MTR benchmark datasets were selected.[1] Table 4, in Appendix A, characterises the datasets regarding the number of samples, input and output variables and also provides a brief description of their content as explored in [19, 25].

4.2 Regression Techniques

In this subsection, we present the two regression techniques employed to build the models: Support Vector Machines (SVM) and Random Forest (RF). We decided to use these algorithms due to their different strategies and wide applications. RF is an ensemble technique where multiple decision trees are combined under the idea of bagging, whereas SVMs relies on statistics foundations to fit a hyperplane able to perform classification and regression. In this research, we used the implementations of the R packages `ranger` and `e1071` for RF and SVM, respectively, along with their default parameter settings.

4.2.1 Random Forest

Originally proposed by Breiman [3], RF is an ensemble algorithm used for both classification and regression tasks. In ensembles algorithms, different models with multiple approaches are employed and, in the case of RFs in specific, decision trees are combined in the form of a forest.

Initially, the algorithm bootstraps by sampling instances with replacement and selecting a subset of features. Following, the subset is used to build a decision tree, repeating this same procedure until a predetermined number of trees are created. After that, in order to perform either classification or regression, all trees are used in conjunction. According to Breiman [3], RF is robust to noise and the decision trees' feature selection capacities are naturally inherited. Moreover, its ensemble properties guarantees no overfitting.

Another main advantage of RFs consists of being able to measure attributes' importance (RF_{imp}). In order to do so, for each sampling performed, about a third of the training set is left aside (out-of-bag examples). Next, out-of-bag examples' attributes are randomly permuted, and tested by the forest. Permutations that increase the RF error can point out which attribute is more relevant. Hence, measuring its importance to the task.

For instance, if the error is increased by inserting noise (permutation) on a specific attribute, such attribute is likely to be critical. The same applies for the opposite situation. In this sense, the RF_{imp} of each variable is calculated by comparing the variation in the ensemble's prediction error obtained before and after the random permutation. If there is no alteration, it implies that the attribute does not contribute to the prediction task. In the same manner, if the error is reduced, the attribute is hindering RF predictions.

For regression, RFs take each trained tree to predict continuous values. In the majority of implementations, the final prediction is given by the mean value predicted by all trees in the forest.

4.2.2 Support Vector Machine

Support Vector Machine is a well established method for signal processing tasks. It was first introduced in Cortes and Vapnik [6] as a classification model for binary problems. SVM objective consists of creating an optimal separation hyperplane that splits the data into two classes. Such hyperplane's boundaries are defined by points referred as support vectors, hence naming the algorithm.

In situations whose data are not linearly separable, kernel tricks are employed. Such operation maps the original data into a higher dimensional space where the problem is more likely to be solvable.

Since we are interested in SVM for regression, we employed Support Vector Regression (SVR) [9]. In SVR, a hyperplane, defined by the Eq. 2, is built by minimising the Eq. 3 subject to Eq. 4.

$$\langle \mathbf{w}, \mathbf{x} \rangle - b = 0 \tag{2}$$

$$\frac{1}{2}\|\mathbf{w}\|^2 \tag{3}$$

$$\begin{cases} y_i - \langle \mathbf{w}, \mathbf{x} \rangle - b \leq \epsilon \\ \langle \mathbf{w}, \mathbf{x} \rangle + b - y_i \leq \epsilon \end{cases} \tag{4}$$

In these Equations, $\mathbf{w}$ corresponds to the normal vector to the hyperplane, x_i is a training sample with label y_i and ϵ is a threshold value for the margin. Hence, all predictions, given by $\langle \mathbf{w}, \mathbf{x} \rangle + b$, must be within the range implied by ϵ. Furthermore, finding the optimal hyperplane is mapped to a convex problem with a feasible solution. Nonetheless, in situations with no feasible solutions, an extra variable ξ may be added to cope with such constraints.

4.3 Performance Evaluation

Aiming at evaluating the models created during the experiments, we used two different metrics: the average

[1] http://mulan.sourceforge.net/datasets-mtr.html

Relative Root Mean Squared Error (aRRMSE) and the Counting of Trained Regression Models (CTRM). Also, the MTR methods were performed using a 10-fold cross-validation strategy to minimise bias.

The aRRMSE is obtained by averaging the RRMSE (Relative Root Mean Squared Error) calculated for each target variable. RRMSE measures a model error decrease over a naive predictor which always outputs the target mean value. The latter servers a baseline in the metric and allows the measurement of the improvement over a shallow predictor, being used in various MTR works [2, 20–23, 25] to compare non-homogeneous targets distributions. The aRRMSE calculation is presented in Eq. 5, where d represents the number of targets variables, N the number of testing instances, y, $\hat{y}$ and $\overline{y}$ denote, respectively, the real value of the output, its predicted and mean values.

$$\text{aRRMSE} = \frac{1}{d}\sum_{t=1}^{d}\sqrt{\frac{\sum_{k=1}^{N}(y_t^k - \hat{y}_t^k)^2}{\sum_{k=1}^{N}(y_t^k - \overline{y}_t)^2}} \tag{5}$$

As the time and memory costs of the MTR local methods directly depends on the employed regressor, we defined the Counting of Trained Regression Models (CTRM) metric to compare how much computational resources they used. CTRM is defined as the total number of ST models induced by an MTR method in a dataset. This metric can be employed as a measure of how much memory each MTR approach uses. In other words, the smaller the CTRM value of an approach, the more memory efficient the evaluated MTR method is. Additionally, training fewer models results in less time complexity of the solution.

Both aRRMSE and CTRM support the comparison of possible method superiority through the application of the Friedman statistical test and the Nemenyi post hoc test with a Critical Difference (CD) diagram, as previously proposed in [7]. They are used to evaluate if the null hypothesis is rejected. The null hypothesis states that the performances of the MTR methods are equivalent regarding the evaluated metric. In this sense, when the null hypothesis is false, we can apply the Nemenyi post hoc test, which states that the performance of two distinct models are significantly different if the corresponding average ranks differ by at least a CD value.

We also defined the *Relative Performance* (RP) and *Relative Size* (RS) measures, derived from aRRMSE and CTRM, respectively. RP and RS are used for evaluating how MOTC compares against ERC in predictive performance and memory efficiency since both methods define chained regression models. RP measures the reduction in error of an MTR approach over another, whereas RS measures how time and memory conservative an MTR method A is when compared with another method B. The RP of a method A against another method B is given according to Eq. 6.

$$RP_{B|A} = \frac{\text{aRRMSE}_A}{\text{aRRMSE}_B} \tag{6}$$

If the obtained RP is equal to one, there is no performance difference between the compared algorithms. An RP greater than one means that B is better than A in the evaluated problem. The same applies to the opposite.

RS calculation is given in Eq. 7. An RS greater than one shows that B spends more regressor models than A. Moreover, an obtained $RS = 0.5$ implies that B spent half the number of models used by A.

$$RS_{B|A} = \frac{\text{CTRM}_A}{\text{CTRM}_B} \tag{7}$$

Lastly, aiming to evaluate the inter-target dependency, we comprised all generated CTs into a single representation. For each dataset, the obtained CTs considering all targets and cross-validation folds were merged into an oriented graph. Every time there was a relation of the type "B explains A", being A and B target variables, the edge weight connecting the two targets was incremented. So, in the obtained graphs the edges represent how many times an output was used to explain another. The resulting structures were called Condensed CT Graphs (CCTG).

5 Results and Discussion

In this section, the comparative results between MOTC, simple ST, and the other MTR approaches (MTRS and ERC) are presented. First, we discuss the results regarding aRRMSE. Then, we discuss how many regression models were trained for each method, while also presenting statistical comparisons concerning both predictive performance and number of induced models. Lastly, our method is compared against ERC, which also uses chained regressors. We show that MOTC had a similar predictive performance compared to ERC, employing fewer regression models in the process. We also present the time complexity analysis for all compared MTR methods. Likewise, some insights on how MOTC could be employed to better comprehend the relationships between targets in an MTR problem are also presented.

5.1 Predictive Performance

Table 1 reports the obtained aRRMSE regarding all MTR methods and base regression techniques[2], per benchmark

[2]The source codes for MOTC and the other evaluated MTR methods are disponible in http://www.uel.br/grupo-pesquisa/remid/?page_id=145.

Table 1 Average Relative Root Mean Squared Error (aRRMSE) results considering all datasets.

Dataset	Algorithm	ST	MTRS	ERC	MOTC
ATP1D	RF	0.3919	0.3904	**0.3902**	0.3910
	SVM	**0.4396**	0.4402	0.4398	0.4398
ATP7D	RF	**0.5164**	0.5169	0.5178	0.5187
	SVM	**0.6404**	0.6414	0.6410	0.6408
OES97	RF	0.5164	0.5135	**0.5133**	0.5153
	SVM	0.6118	0.6123	**0.6109**	0.6116
OES10	RF	**0.4070**	0.4081	**0.4070**	0.4073
	SVM	0.5464	0.5456	**0.5451**	0.5464
RF1	RF	0.0782	**0.0582**	0.0731	0.0723
	SVM	0.1215	**0.1070**	0.1151	0.1191
RF2	RF	0.0847	**0.0784**	0.0852	0.0872
	SVM	0.1095	**0.1064**	0.1084	0.1091
SCM1D	RF	0.2871	**0.2757**	0.2823	0.2823
	SVM	0.3309	**0.3232**	0.3258	0.3260
SCM20D	RF	0.3647	0.3313	0.3347	**0.3277**
	SVM	0.3972	0.3522	0.3474	**0.3472**
SDM	RF	0.6721	0.6631	0.6661	**0.6620**
	SVM	0.7699	0.7714	0.7667	**0.7654**
SF1	RF	**1.0051**	1.1280	1.0161	1.0578
	SVM	0.9390	0.9477	**0.9250**	0.9337
SF2	RF	**0.8487**	0.9410	0.8617	0.8729
	SVM	**0.7825**	0.7787	0.7827	0.7843
Jura	RF	0.6061	0.5974	**0.5969**	0.5971
	SVM	0.6409	0.6413	**0.6391**	0.6418
WQ	RF	0.9066	0.9392	**0.9059**	0.9139
	SVM	0.9630	**0.9535**	0.9581	0.9617
ENB	RF	0.1504	**0.1173**	0.1304	0.1206
	SVM	0.2499	**0.2173**	0.2404	0.2322
Slump	RF	0.8365	0.8325	**0.8222**	0.8497
	SVM	0.6924	0.6862	0.6818	0.7056
Andro	RF	0.7941	**0.7349**	0.7614	0.7734
	SVM	1.1348	**0.9243**	1.0089	1.0208
OSALES	RF	0.7577	**0.7275**	0.7332	0.7455
	SVM	1.1726	**1.1685**	1.1702	1.1716
SCPF	RF	0.9263	0.9387	0.8778	**0.8711**
	SVM	0.8242	0.8256	**0.8151**	0.8159

dataset. The columns ST, MTRS, ERC and MOTC represent the MTR approaches, while the base regressors are presented in column Algorithm, resulting in a base regressor-MTR approach combination by line. The highlighted value of aRRMSE corresponds to the better performing algorithm/MTR approach pair for that dataset.

MTRS performed better in 13 algorithm-MTR approach combinations (six cases using RF and the remaining with SVM as base regressor). ERC achieved the best prediction errors in 11 cases (six times when using RF and five with SVM). ST got the best aRRMSE in seven cases (four with the RF and three with SVM). MOTC, obtained the best results in five cases (three times combined with RF and two cases with SVM). Although our approach was out-performed by the others, the performance difference is minimal, being very close to ERC in the majority of the cases. Nonetheless, MOTC outperforms ERC, which is also based on chained regressors, by a large margin when considering memory and time, as discussed further in this work. Also, our method offers a highly representative model of how targets are correlated to each other. None of the compared approaches (ST, MTRS and ERC) can be used to explain the existing dependencies between the output variables.

5.2 Number of Induced Models

We report the mean CTRM value for each MTR method in Table 2. This analysis is particularly relevant, since the number of generated models directly impacts the obtained solution's time and memory costs. Since one of our goals was to obtain a memory and time conservative method, without compromising prediction performance, this analysis is a core part of this paper.

Table 2 Mean Counting of Trained Regression Models (CTRM) obtained by each method in all evaluated datasets.

Dataset	ST	MTRS	ERC	MOTC (RF)	MOTC (SVM)
ATP1D	6	12	60	30	30
ATP7D	6	12	60	39.9	37.9
OES97	16	32	160	84.8	87.1
OES10	16	32	160	84.1	81.8
RF1	8	16	80	24.3	24.3
RF2	8	16	80	24.3	24.3
SCM1D	16	32	160	64	64.5
SCM20D	16	32	160	65.2	65.2
EDM	2	4	4	4	4
SF1	3	6	18	8.6	8.4
SF2	3	6	18	7.3	7.3
Jura	3	6	18	9.1	9.1
WQ	14	28	140	64	63.5
ENB	2	4	4	4	4
Slump	3	6	18	9.4	9.4
Andro	6	12	60	30.6	30
OSALES	12	24	120	51.5	52.6
SCFP	3	6	18	9	9

ST, as the simplest approach, induces as many regressor models as the number of problem targets. MTRS always generates twice the number of ST models. In its turn, ERC creates the maximum number of combinations of targets if the number of outputs is lesser than three. Otherwise, ten random combinations are chosen, so CTRM is equal to ten times the number of target variables. MOTC adaptively chooses the number of trained predictors, which is equal to the summed number of nodes in all generated CTs. The choice of MOTC's $\text{Max}_{\text{depth}}$ parameter and the application of HB directly impact the obtained CTRM.

As expected, ST is the most time and memory conservative approach, followed by MTRS in all cases except when the number of targets is equal to two. In these cases, both ERC and MOTC are tied with MTRS. MOTC is the third most conservative approach, independently of the chosen regressor, and at the least position comes ERC. Despite employing more models than the ST and MTRS approaches, MOTC dynamically adapts its CT structures for each problem offering insights on how the targets influence each other. This fact offers a useful tool for experts and non-experts when dealing with MTR tasks.

The slight different CTRM obtained by MOTC when using RF or SVM as base regressors is a result of the target importance metric choice. We calculated RF_{imp} upon a set of RF models trained using only the target variables. RF's variable importance value takes into consideration the increase in ensemble's predictive error brought by random permuting the values of an explaining feature. Also, each tree in a RF is constructed by sampling training instances and subsetting features at random. In fact, there are many aspects of randomness which result in slightly different CTs, depending on the number of elements and targets of the problem, as well as the number of trees used to calculate RF_{imp}.

The authors encourage the evaluation of the impact brought by using different importance metrics and values for CT's $\text{Max}_{\text{depth}}$ on the obtained CTRM and aRRMSE. This analysis, however, is out of the scope of this work.

5.3 Statistical Comparison

We performed the Friedman test on the predictive performance of the compared MTR methods. Figure 2 reports the Nemenyi post test ranked comparison considering the predictive performance obtained over all datasets. Lower rankings mean better predictive performance (with CD $=$ 0.7837 and $\alpha = 0.05$). The obtained Friedman test's p-value ($< 2.2e-16$) confirm there were statistical differences between compared MTR methods and employed regressors, regarding the obtained aRRMSE.

As shown in Fig. 2, the first connected group comprehends to ERC, MTRS and MOTC using RF as the base regressor model. The second group comprehends MOTC and ST, again along with RF. In this sense, there is no statistical difference between ERC, MTRS and MOTC. In the third connected group, corresponding to using SVM as regressor base model, MTRS appears at the first position, followed by ERC, MOTC and ST. Again, no statistical difference was observed between ERC, MTRS and MOTC. Using both RF and SVM, MOTC is connected with ST. It seems that the chosen base regression algorithm can impact in the ranking of the selected MTR methods. Besides, the distance observed between MTR approaches was lesser when using RF, which was related to the greater generalisation capacity commonly achieved by ensemble methods.

As discussed in Section 5.4, MOTC performed very comparably to ERC in almost all cases. This fact is related to their similar resulting models' structure. In most of the

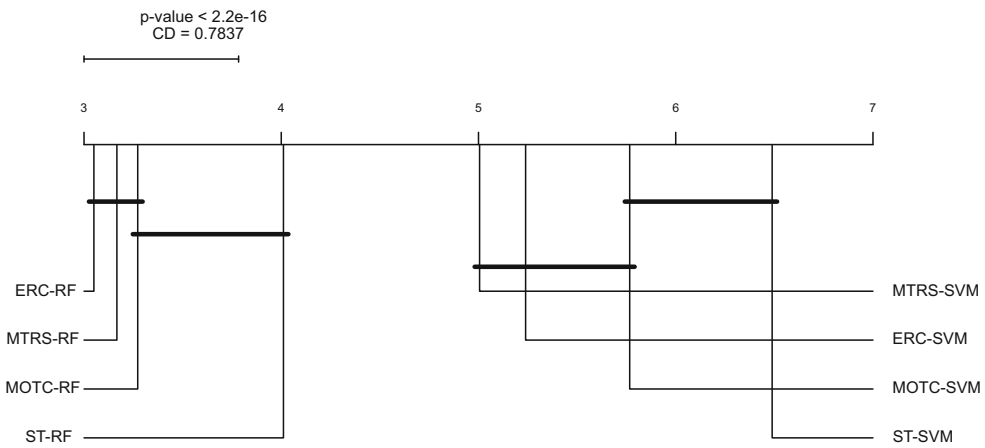


Figure 2 Nemenyi post test resulting, considering the predictive performance over all datasets.

cases, MOTC was equivalent or obtained slightly greater error than ERC. For this reason, when MTRS outperformed ERC, it also was superior to MOTC. On another hand, when ERC achieved the least aRRMSE, MOTC results were very close to it, but still somewhat worse. Thus, MOTC appears at third position as a reflex the few cases it outperformed both ERC and MTRS.

We present the statistical comparison regarding the number of generated models per MTR method (at $CD = 0.4555$ and $\alpha = 0.05$) in Fig. 3. As expected, ST provides the most conservative approach, being at the first position in all cases. MTRS comes in second place, spending twice the ST number of models. MOTC is the third more conservative approach, independently of the chosen regressor. As expected, ERC is the worst method, expending a lot more regression models than the other MTR approaches. This scenario could be even worse if one chooses to remove the number of generated chains limitation, proposed in original ERC formulation (maximum number of random permutations equal to ten). It is worth mentioning that all methods were statistically different among themselves in this test.

We also would like to emphasise that MOTC adaptively generates a different number of models by analysing how the problem targets relate to each other. All compared models (ST, MTRS and ERC) produce a fixed number of regression models, which are not easy to interpret in the inter-target relationship context. In contrast, MOTC offers clear information about how an output helps to explain the distribution of the other targets. Further, by using the CTs from MOTC, it is possible to compare their results with prior knowledge about a research field and even discover new interesting properties and inferences from the targets.

Table 3 Relative Performance (RP) and Relative Size (RS) when comparing MOTC with ERC in all datasets.

Dataset	$RP_{MOTC\|ERC}$		$RS_{MOTC\|ERC}$	
	RF	SVM	RF	SVM
ATP1D	1.00	1.00	0.50	0.50
ATP7D	1.00	1.00	0.67	0.63
OES97	1.00	1.00	0.53	0.54
OES10	1.00	1.00	0.53	0.51
RF1	1.01	0.96	0.30	0.30
RF2	0.95	0.99	0.30	0.30
SCM1D	1.00	1.00	0.40	0.40
SCM20D	1.02	1.00	0.41	0.41
EDM	1.01	1.00	1.00	1.00
SF1	0.97	0.99	0.48	0.47
SF2	0.99	1.00	0.41	0.41
Jura	1.00	1.00	0.51	0.51
WQ	0.99	1.00	0.46	0.45
ENB	1.09	1.03	1.00	1.00
Slump	0.97	0.97	0.52	0.52
Andro	1.02	1.00	0.51	0.50
OSALES	0.98	1.00	0.43	0.44
SCFP	1.01	1.00	0.50	0.50

5.4 MOTC and ERC Comparison

The idea of chaining regressor models is shared by MOTC and ERC, but the second does this action in a naive strategy. Table 3 presents the obtained RP and RS when comparing MOTC with ERC.

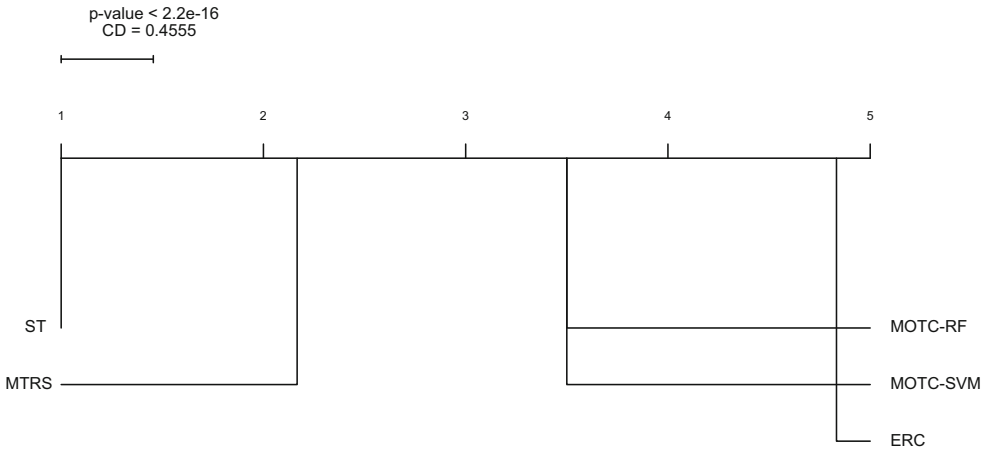


Figure 3 Nemenyi post test resulting, considering the generated number of models in all datasets.

MOTC got RP values very close to ERC in the majority of datasets. In the worst case, MOTC achieved 95% of ERC's performance; in some cases our approach stood out.

On the other hand, when it comes to RM, MOTC was prominently superior to ERC. There were only two cases where MOTC generated as many models as ERC. They correspond to datasets EDM and ENB, which have only two target variables. Excluding these simpler problems, in the worst case, MOTC built 65% fewer models than ERC. In fact, in most of the time, MOTC used about half regressors than ERC did.

Therefore, our proposed approach obtained a competitive predictive performance when compared with ERC. Furthermore, it combines modest time and memory costs for modelling through an informative and easy to interpret structure.

5.5 Complexity Analysis

Consider d as the number of problem targets and b, the complexity of the chosen base regressor. ST induces as many models as the number of outputs, so its time complexity is $\mathcal{O}(d \times b)$. MTRS, in its turn, produces twice the number of ST's models. For that reason, MTRS's time complexity is $\mathcal{O}(2 \times d \times b)$, which does not asymptotically differ from ST. ERC, without limiting settings, produces as many regressors as the maximum number of targets permutations. In this sense, ERC's complexity would be $\mathcal{O}(d^2 \times (d-1)! \times b)$. However, by following the ERC's authors recommendation of generating only ten random permutations if the number of targets is greater than three, the obtained complexity is $\mathcal{O}(10 \times d \times b)$.

MOTC generates regression models with a tree fashion. The growing of new nodes in the produced CTs depends on two main factors: the max tree depth constraint (parameter $\max_{\text{depth}}$) and the branching factor. The latter is determined by using or not the HBs. Suppose ζ to be the summed number of nodes contained in all targets CTs, excluding the redundant leaf nodes, as previously discussed in Section 3. MOTC time complexity is $\mathcal{O}(\zeta \times b)$. Although there is no asymptotic difference between MOTC and constrained ERC version, our experiments proved that the former method is the most time and memory efficient.

Also, when the number of targets is sufficiently small, MOTC generates as many models as MTRS or ERC. In fact, if the $\max_{\text{depth}}$ is setted to one, MOTC will mimic MTRS. Moreover, if MOTC branching factor get too close to d and/or the maximum depth of CTs is sufficiently large, our method should generate more regression models than ERC. Nevertheless, as our experiments demonstrated, MOTC with its default parameters is capable of achieving comparable performance results while being more time and memory conservative than ERC.

5.6 Analysis of the Relationship Between Targets with CTs

The generated CCTGs allow us to get some interesting information. For example, one could analyse that an output almost always helps to explain another, but the opposite is not true. Also, the presence of connections among targets with small edge values is an indication of little influence of a target on another. Additionally, cycles inside the graphs could be detected, meaning there are indirect relationships among targets. Lastly, different groups of dependency could be detected, resulting in decomposing a MTR target into smaller multi-output tasks.

All obtained CCTGs are presented in Appendix B "*Obtained condensed chaining tree graphs and target labels*". For the matter of readability, in all graphs, the target names were substituted by T followed by an output identifier. One could check their original names by referring to Tables 5 to 11.

We present some real examples about the mentioned CCTG behaviour. Some cases were pretty straightforward, presenting mutual dependencies in all cases, like in datasets EDM and ENB (Fig. 4). They represent the two datasets with the smallest number of target variables, so it is to be expected a simple inter-target relationship.

Other interesting cases were discovered when analysing the datasets SCPF and Slump (Fig. 5). In SCPF, it is possible to observe that there are no dependencies between targets T2 (num_views) and T1 (num_comments). Meanwhile, both of them influence and are influenced by T3 (num_votes). As highlighted in Section 4.1, SCPF evaluates the number of votes, views and comments obtained by certain online issues within a period. Our analysis shows that the number of votes that an issue receives depends on its number of views and comments, which seems reasonable in the context of online content resources. However, an issue could achieve high rates of views and comments, whereas not being voted.

The Slump dataset concerns the prediction of three properties of concrete (slump, flow and compressive strength). As shown in Fig. 5, there are mutual relationships between targets T2 (Flow) and T3 (Slump), and both of them influence the Compressive Strength (T1) of concrete. However, they are not influenced by T1. MOTCs takes advantage of not inserting T1 as additional prediction feature when modelling T2 and T3.

In addition, another behaviour was observed when combining the generated CTs. There were cases where the targets were grouped regarding their inter-dependencies. In these situations, targets within a group did not influence outputs outside it. In this sense, this behaviour indicated that an original MTR task could be decomposed into smaller problems. The creation of separate target groups was observed in datasets andro, ATP1D (Fig. 6), SCM1D and SCM20D (Fig. 11).

Take as an example the dataset ATP1D. It concerns the prediction of the minimum air ticket price in the next day, given an observed date. ATP1D targets correspond to the expected minimum price for: (1) any airline with any number of stops, (2) any airline non-stop only, (3) Delta Airlines, (4) Continental Airlines, (5) Airtrain Airlines, and (6) United Airlines. Figure 6 give us pieces of evidence that the next day minimum price for non-stop airlines have influenced and was influenced by the price of Continental Airlines. The same comparison applies when comparing Delta and Airtrain Airlines, and United Airlines with flights with any number of stops. On the other hand, no influence was observed among different groups, considering all generated CTs and cross-validation data partitions.

Furthermore, complex patterns of relationships were observed in most of the datasets whose number of outputs are greater than eight. As the number of targets has increased, also has risen the diversity of the dependencies between the output variables. In datasets OES10 (Fig. 7), OES97 (Fig. 8), RF1, RF2 (Fig. 10), OSALES (Fig. 9), and WQ (Fig. 12), diverse inter-target dependence patterns were observed. In these datasets, most of the previously discussed cases of influence can be seen. In contrast, there were clear and simple patterns of relationship among targets in datasets SCM1D and SCM20D (Fig. 11), as previously presented. Thus, there is no direct association between the number of outputs in a predictive problem and the complexity level of the statistical dependencies between targets.

Lastly, by using a CCTG representation, one could apply a threshold value to consider the most prominent cases of target influence. In this way, by removing edges with small weights, the complex relationship between outputs could be decomposed into simpler MTR tasks or even separate ST problems. In fact, in many cases the weights in edges were unbalanced. For instance, a proper thresholding scheme would separate targets T1 and T4 into a separate MTR problem, when considering datasets RF1 and RF2 (Fig. 10). The RF datasets deal with the prediction of the Mississipi river flow in eight different sites. The strong dependency between sites CHSI2_48H_0 (T1) and EADM7_48H_0 (T4) should reflect some physical, geological and positional aspects which relate these two measured locals.

6 Conclusion

This paper proposed a new method, called Multi-output Tree Chaining (MOTC), for solving multi-target regression problems. Our solution describes the relationships between target variables as a tree structure toward a CT. In this structure, each node represents an output, and its children denote the targets which most explain it. CTs take into consideration a target relevance measure to assess the dependencies between outputs. We employed the Random Forest variable importance metric as a nonlinear descriptor of inter-target relations. Also, when constructing the CTs, aiming at limiting their ramification factor, we employed the Hoeffding Bound only to select the targets which most explain the targeted output. After building a CT for each target, regression models were inducted for each node, starting from the leaves. In MOTC, intermediate nodes use the original features of the problem augmented by the predictions that come from their descendants models.

An extensive experimental setup consisting in 18 standard multi-target benchmark datasets, combined with two regression techniques (Random Forest and Support Vector Machine), where we compared MOTC with three state-of-art MTR methods (ST, MTRS and ERC) was performed. The results showed that MOTC obtained competitive predictive performance. In fact, the performed statistical tests demonstrated that MOTC has no significant difference from MTRS and ERC methods concerning predictive performance, being superior to ST. It is worth mentioning that the choice of the used regression technique proved to impact in the final performance of the multi-target regression methods. This fact should be analysed in future works. Nevertheless, our method presented significant reduction over ERC regarding memory and time complexity.

Besides that, MOTC offers a comprehensive representation to analyse multi-target tasks, which none of the existing solutions present, enabling the verification of prior knowledge about a problem, and even the discovering of new properties.

Acknowledgements The authors would like to thank CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) and FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) for financial support.

Appendix A: Datasets Used in the Experiments

Table 4 Dataset's characteristics: dataset name, number of examples, number of input variables, number of targets, and description.

Dataset	#Examples	#Input	#Outputs	Description
ATP1D	337	411	6	Minimum air ticket price in the next day for different options of companies and number of stops.
ATP7D	296	411	6	Minimum air ticket price over the next seven days for different options of companies and number of stops.
OES97	334	263	16	Occupational employment surveys performed in 1997, containing the estimated number of employees in different jobs.
OES10	403	298	16	Occupational employment surveys performed in 2010, containing the estimated number of employees in different jobs.
RF1	9125	64	8	River flows in the next 48 hours in eight different rivers in the Mississippi River network.
RF2	9125	576	8	River flows in the next 48 hours in eight different rivers in the Mississippi River network with precipitation forecast information added to the input variables.
SCM1D	9803	280	16	Prices of different products in the next day in a supply chain management.
SCM20D	8966	61	16	Mean price of different products over the next 20 days in a supply chain management.
EDM	154	16	2	Parameter settings controlled by a human operator during electrical discharge machining.
SF1	323	10	3	Number of times common, moderate and severe solar flares were observed in a 24h interval.
SF2	1066	10	3	Number of times common, moderate and severe solar flares were observed in a 24h interval.
Jura	359	15	3	Concentration of heavy metals in the top soil of a region in Swiss called Jura.
WQ	1060	16	14	Relative representation of plant and animal species in Slovenian rivers based on physical and chemical water quality parameters.
ENB	768	8	2	Heating and cooling load required for energy efficient buildings based on their dimensions and material choice.
Slump	103	7	3	Slump, flow and compressive strength concrete properties based on the amount of different concrete ingredients.
Andro	49	30	6	Water quality attributes in Thermaikos Gulf of Thessaloniki, Greece.
OSALES	639	413	12	Online monthly sales in the first twelve months after a product launch.
SCFP	1137	23	3	Number of views, clicks and comments on online issues.

Appendix B: Obtained Condensed Chaining Tree Graphs and Target Labels

Table 5 Targets' labels for datasets EDM, ENB and Jura.

Label	EDM	ENB	Jura
T1	DFlow	Y1	Cd
T2	DGap	Y2	Co
T3	–	–	Cu

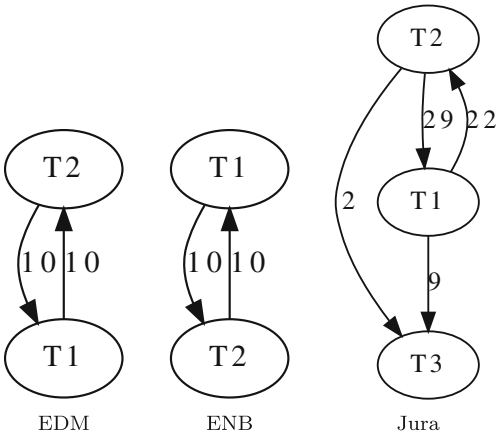


Figure 4 CCTG for datasets EDM, ENB and Jura.

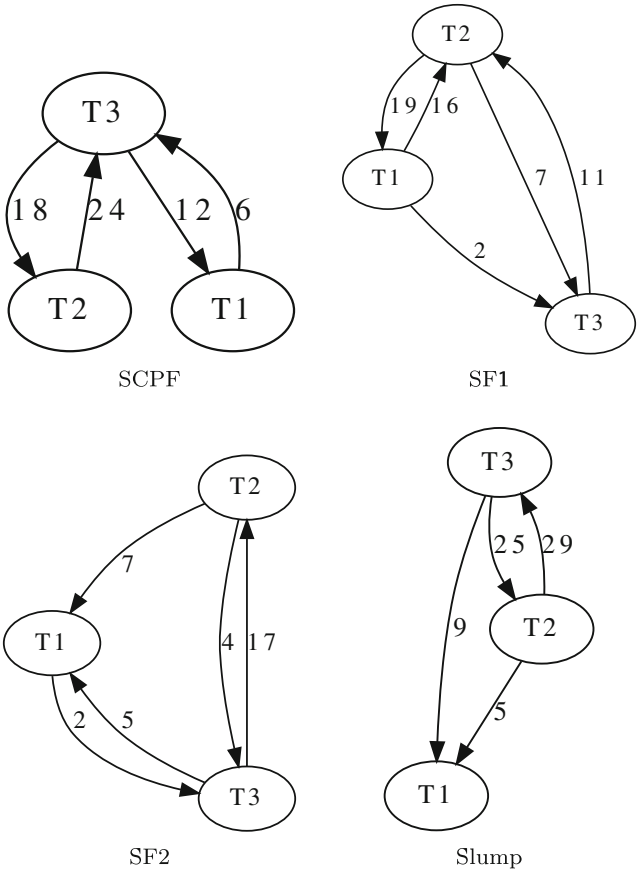


Figure 5 CCTG for datasets SCPF, SF1, SF2 and Slump.

Table 6 Targets labels for datasets SCPF, SF1, SF2 and Slump.

Label	SCPF	SF1	SF2	Slump
T1	num_comments	c-class	c-class	Compressive_Strength_Mpa
T2	num_views	m-class	m-class	FLOW_cm
T3	num_votes	x-class	x-class	SLUMP_cm

Table 7 Targets' labels for datasets Andro, ATP1D and ATP7D.

Label	Andro	ATP1D	ATP7D
T1	Target	LBL+aCOminpA+fut_001	LBL+aCOminpA+bt7d_000
T2	Target_2	LBL+aDLminpA+fut_001	LBL+aDLminpA+bt7d_000
T3	Target_3	LBL+aFLminpA+fut_001	LBL+aFLminpA+bt7d_000
T4	Target_4	LBL+ALLminp0+fut_001	LBL+ALLminp0+bt7d_000
T5	Target_5	LBL+ALLminpA+fut_001	LBL+ALLminpA+bt7d_000
T6	Target_6	LBL+aUAminpA+fut_001	LBL+aUAminpA+bt7d_000

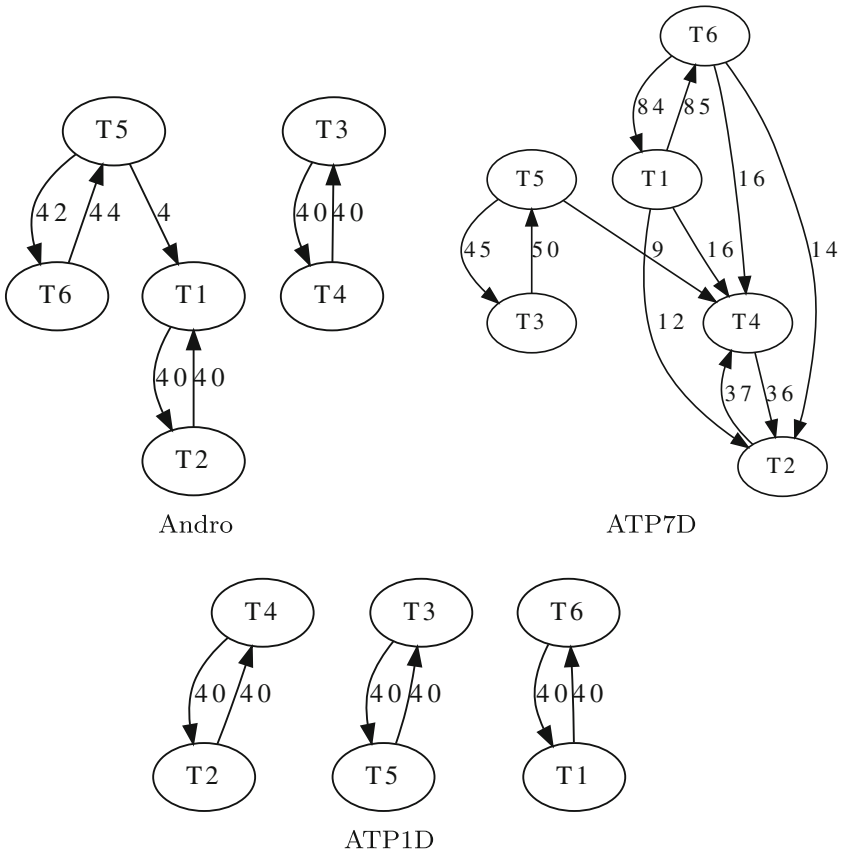


Figure 6 CCTG for datasets Andro, ATP1D and ATP7D.

Table 8 Targets' labels for dataset OES10.

Label	OES10
T1	119032_Education_Administrators_Elementary_and_Secondary_School
T2	151131_Computer_Programmers
T3	151141_Database_Administrators
T4	172141_Mechanical_Engineers
T5	291051_Pharmacists
T6	291069_Physicians_and_Surgeons__All_Other
T7	291127_Speech-Language_Pathologists
T8	292037_Radiologic_Technologists_and_Technicians*
T9	292071_Medical_Records_and_Health_Information_Technicians
T10	392021_Nonfarm_Animal_Caretakers
T11	412021_Counter_and_Rental_Clerks
T12	419022_Real_Estate_Sales_Agents
T13	431011_First-Line_Supervisors_of_Office_and_Administrative_Support_Workers
T14	432011_Switchboard_Operators_Including_Answering_Service
T15	513021_Butchers_and_Meat_Cutters
T16	519061_Inspectors_Testers_Sorters_Samplers_and_Weighers

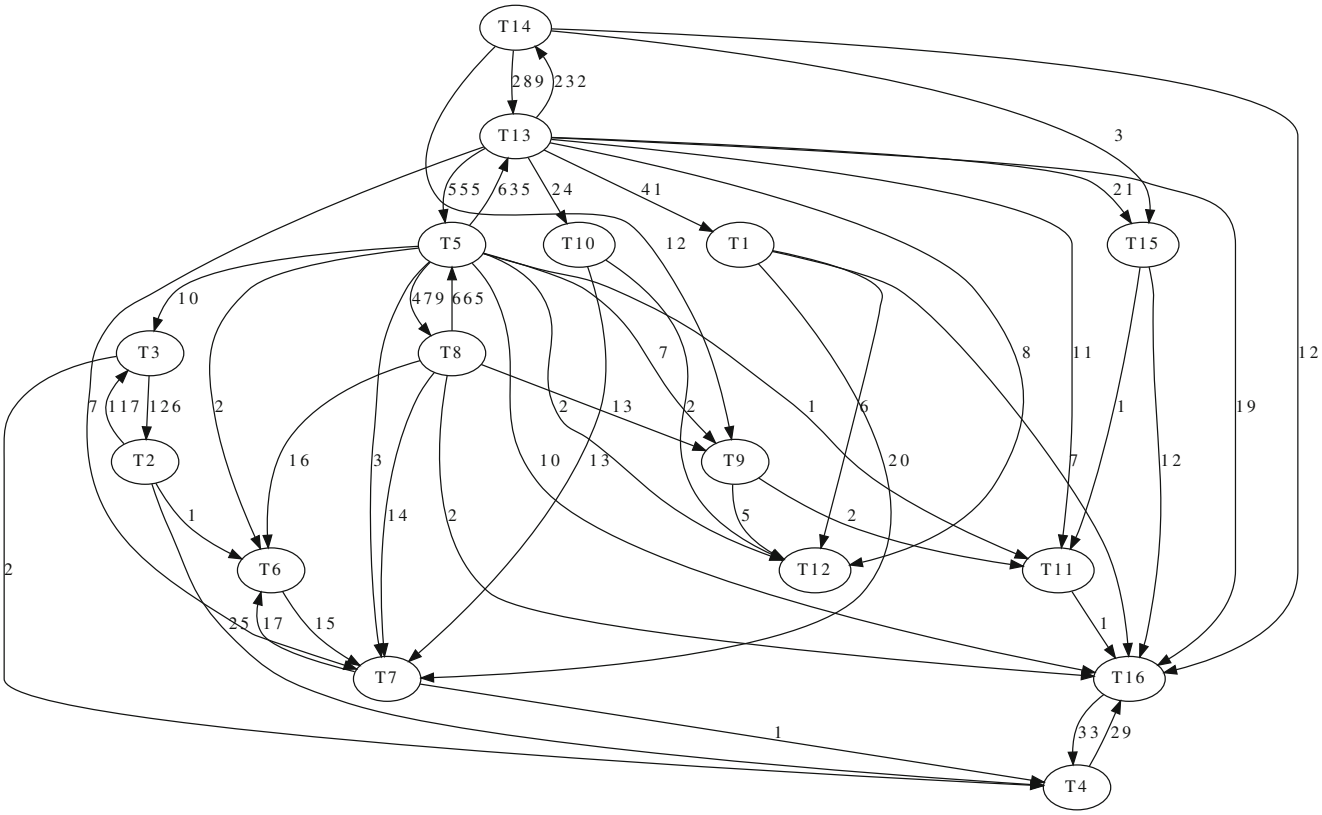


Figure 7 CCTG for dataset OES10.

Table 9 Targets' labels for dataset OES97.

Label	OES97
T1	13008_Purchasing_Managers
T2	15014_Industrial_Production_Managers
T3	15017_Construction_Managers
T4	21114_Accountants_and_Auditors
T5	27108_Psychologists
T6	27311_Recreation_Workers
T7	32314_Speech-Language_Pathologists_and_Audiologists
T8	32511_Physician_Assistants
T9	53905_Teacher_Aides_and_Educational_Assistants_Clerical
T10	58028_Shipping_Receiving_and_Traffic_Clerks
T11	65032_Cooks_Fast_Food
T12	85110_Machinery_Maintenance_Mechanics
T13	92965_Crushing_Grinding_Mixing_and_Blending_Machine_Operators_and_Tenders
T14	92998_All_Other_Machine_Operators_and_Tenders
T15	98502_Machine_Feeders_and_Offbearers
T16	98902_Hand_Packers_and_Packagers

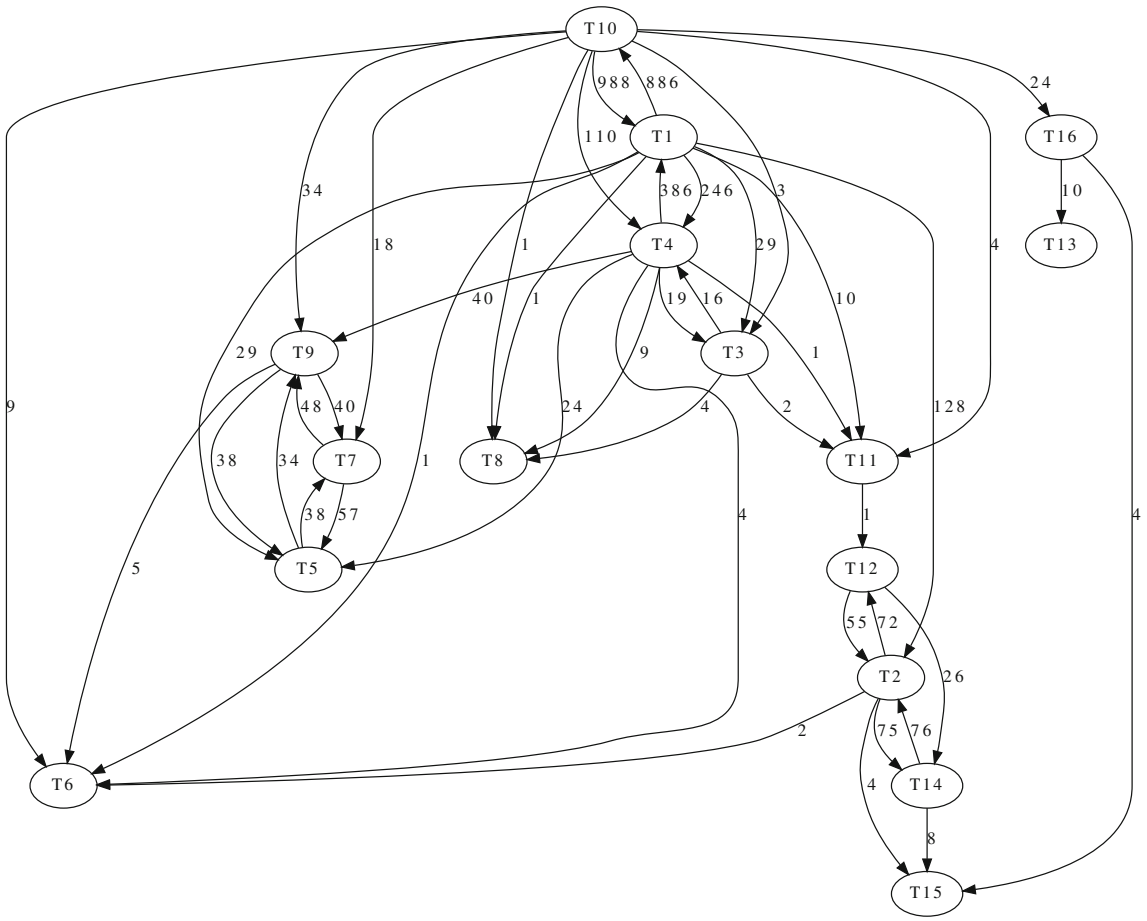


Figure 8 CCTG for dataset OES97.

Table 10 Targets labels for datasets OSALES, RF1 and RF2.

Label	OSALES	RF1	RF2
T1	Outcome_M1	CHSI2_48H_0	CHSI2_48H_0
T2	Outcome_M2	CLKM7_48H_0	CLKM7_48H_0
T3	Outcome_M3	DLDI4_48H_0	DLDI4_48H_0
T4	Outcome_M4	EADM7_48H_0	EADM7_48H_0
T5	Outcome_M5	NAPM7_48H_0	NAPM7_48H_0
T6	Outcome_M6	NASI2_48H_0	NASI2_48H_0
T7	Outcome_M7	SCLM7_48H_0	SCLM7_48H_0
T8	Outcome_M8	VALI2_48H_0	VALI2_48H_0
T9	Outcome_M9	–	–
T10	Outcome_M10	–	–
T11	Outcome_M11	–	–
T12	Outcome_M12	–	–

Table 11 Targets labels for dataset SCM1D, SCM20D and WQ.

Label	SCM1D	SCM20D	WQ
T1	LBL	LBL	17300
T2	MTLp2	MTLp2A	19400
T3	MTLp3	MTLp3A	25400
T4	MTLp4	MTLp4A	29600
T5	MTLp5	MTLp5A	30400
T6	MTLp6	MTLp6A	33400
T7	MTLp7	MTLp7A	34500
T8	MTLp8	MTLp8A	37880
T9	MTLp9	MTLp9A	38100
T10	MTLp10	MTLp10A	49700
T11	MTLp11	MTLp11A	50390
T12	MTLp12	MTLp12A	55800
T13	MTLp13	MTLp13A	57500
T14	MTLp14	MTLp14A	59300
T15	MTLp15	MTLp15A	–
T16	MTLp16	MTLp16A	–

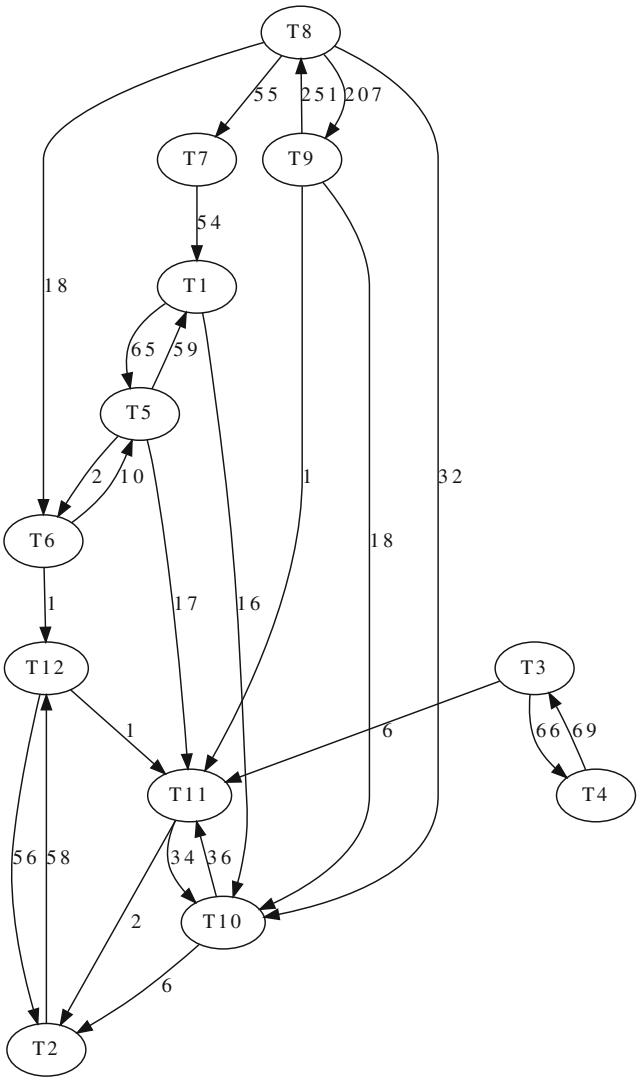


Figure 9 CCTG for dataset OSALES.

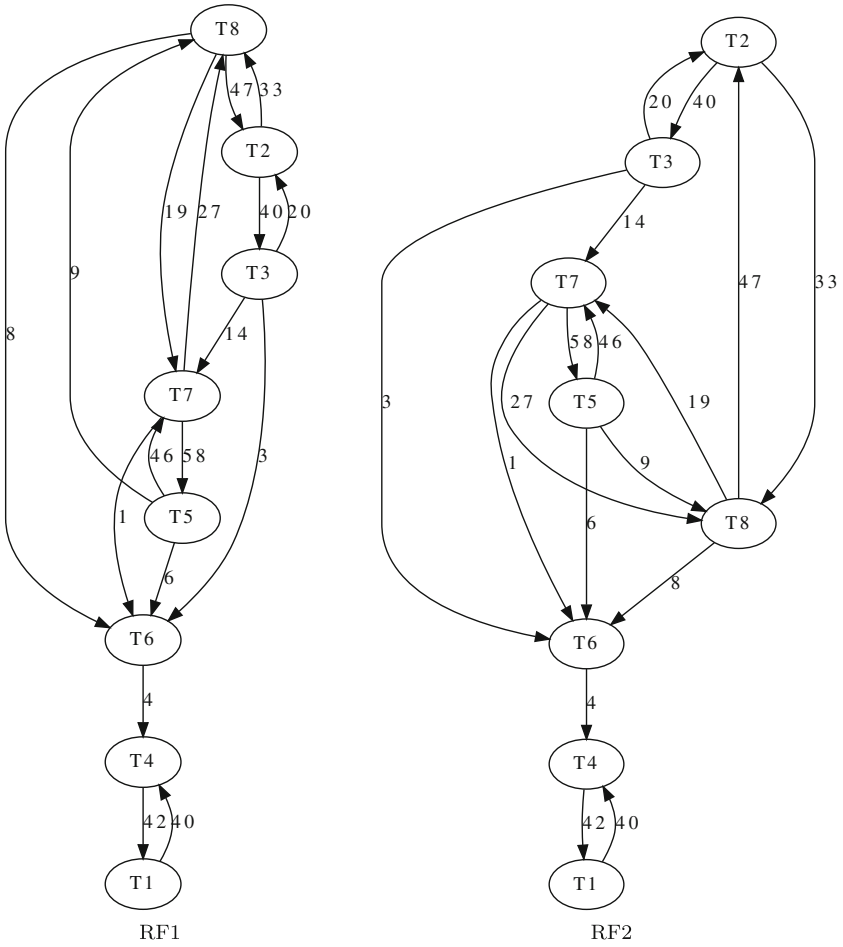


Figure 10 CCTG for datasets RF1 and RF2.

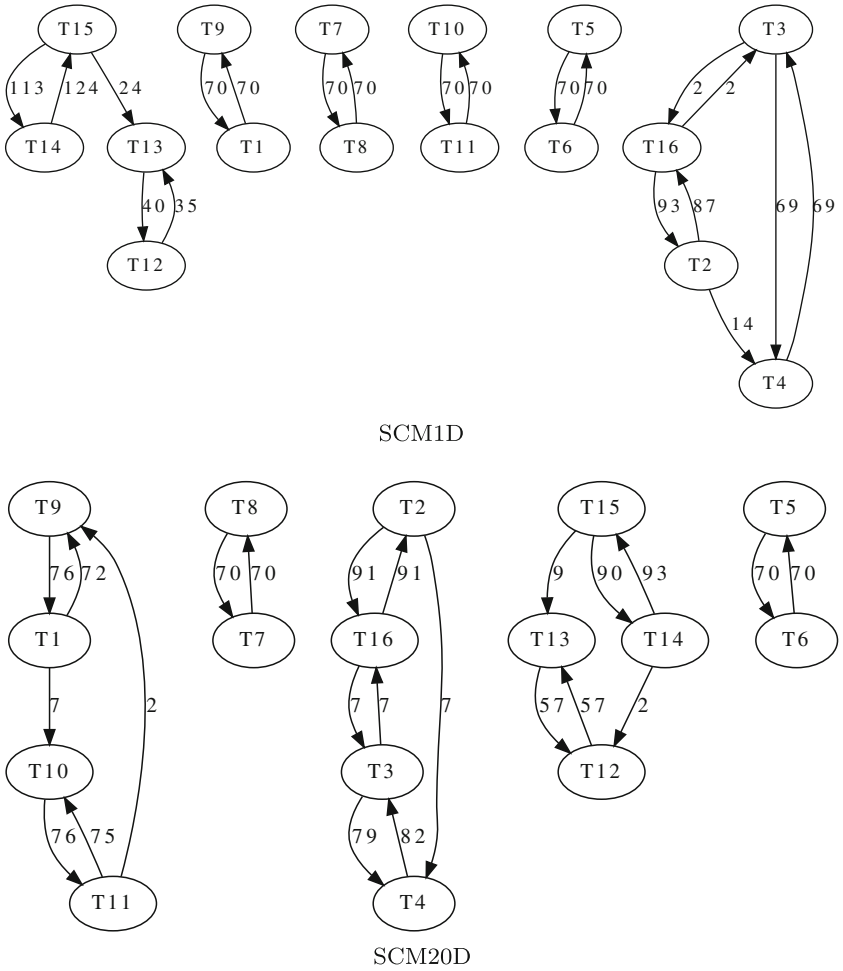


Figure 11 CCTG for datasets SCM1D and SCM20D.

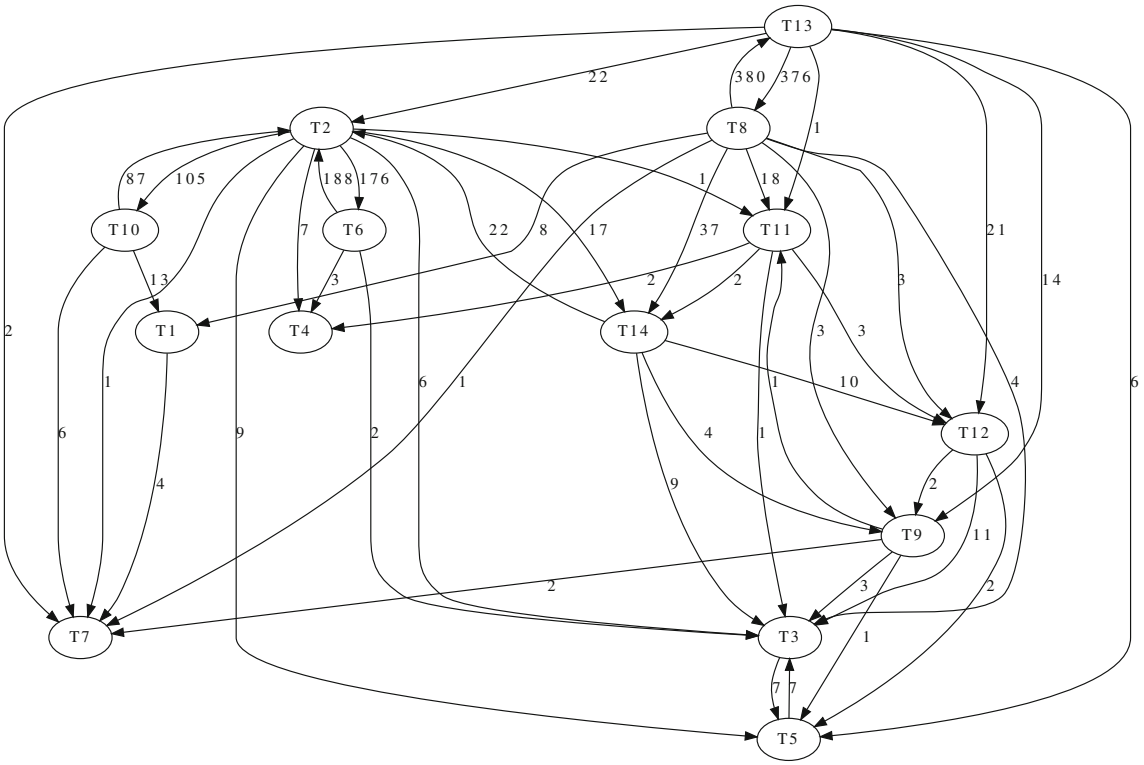


Figure 12 CCTG for dataset WQ.

References

1. Aho, T., Zenko, B., Dzeroski, S., Elomaa, T. (2012). Multi-target regression with rule ensembles. *Journal of Machine Learning Research*, *13*, 2367–2407.
2. Borchani, H., Varando, G., Bielza, C., Larrañaga, P. (2015). A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *5*(5), 216–233.
3. Breiman, L. (2001). Random forests. *Machine learning*, *45.1*, 5–32. https://doi.org/10.1017/CBO9781107415324.004.
4. Brugger, D., Rosenstiel, W., Bogdan, M. (2011). Online SVR training by solving the primal optimization problem. *Journal of Signal Processing Systems*, *65*(3), 391–402.
5. Chen, H., & Ser, W. (2011). Sound source DOA estimation and localization in noisy reverberant environments using least-squares support vector machines. *Journal of Signal Processing Systems*, *63*(3), 287–300.
6. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1023/A:1022627411411.
7. Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, *7*, 1–30.
8. Di Persio, L., & Honchar, O. (2016). Artificial neural networks architectures for stock price prediction: comparisons and applications. *International Journal of Circuits, Systems and Signal Processing*, *10*, 403–413.
9. Drucker, H., Burges, C.J.C., Kaufman, L., Smola, A.J., Vapnik, V. (1997). Support vector regression machines. In Mozer, M.C., Jordan, M.I., Petsche, T. (Eds.) *Advances in neural information processing systems* (Vol. 9, pp. 155–161). MIT Press. http://papers.nips.cc/paper/1238-support-vector-regression-machines.pdf.
10. Evgeniou, T., Figueiras-Vidal, A.R., Theodoridis, S. (2008). Emerging machine learning techniques in signal processing.
11. Gama, J., & Brazdil, P. (2000). Cascade generalization. *Machine Learning*, *41*(3), 315–343. https://doi.org/10.1023/A:1007652114878.
12. Genuer, R., Poggi, J.M., Tuleau-Malot, C. (2010). Variable selection using random forests. *Pattern Recognition Letters*, *31*(14), 2225–2236. https://doi.org/10.1016/j.patrec.2010.03.014. http://www.sciencedirect.com/science/article/pii/S0167865510000954.
13. Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, *58*(301), 13–30. https://doi.org/10.1080/01621459.1963.10500830. http://amstat.tandfonline.com/doi/abs/10.1080/01621459.1963.10500830.
14. Katagiri, S., Nakamura, A., Adali, T., Tao, J., Larsen, J., Tan, T. (2014). Guest editorial: Machine learning for signal processing. *Journal of Signal Processing Systems*, *74*(3), 281–283. https://doi.org/10.1007/s11265-014-0871-6.
15. Kocev, D., Džeroski, S., White, M.D., Newell, G.R., Griffioen, P. (2009). Using single- and multi-target regression trees and ensembles to model a compound index of vegetation condition. *Ecological Modelling*, *220*(8), 1159–1168.

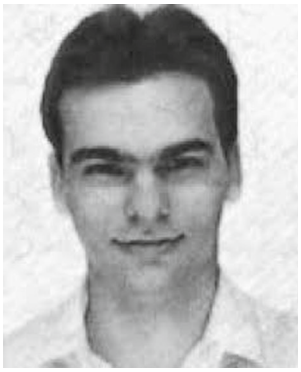
16. Kocev, D., Vens, C., Struyf, J., Džeroski, S. (2007). Ensembles of multi-objective decision trees. In *European conference on machine learning* (pp. 624–631). Springer.
17. Kocev, D., Vens, C., Struyf, J., Džeroski, S. (2013). Tree ensembles for predicting structured outputs. *Pattern Recognition*, *46*(3), 817–833.
18. Li, X., & Zheng, J. (2016). Active learning for regression with correlation matching and labeling error suppression. *IEEE Signal Processing Letters*, *23*(8), 1081–1085.
19. Lichman, M. (2013). UCI machine learning repository. http://archive.ics.uci.edu/ml.
20. Mastelini, S.M., Santana, E.J., Cerri, R., Barbon, S. Jr. (2017). DSTARS: a multi-target deep structure for tracking asynchronous regressor stack. In *Brazilian conference on intelligent systems. BRACIS 2017*.
21. Melki, G., Cano, A., Kecman, V., Ventura, S. (2017). Multi-target support vector regression via correlation regressor chains. *Information Sciences*, *415*, 53–69.
22. Moyano, J.M., Gibaja, E.L., Ventura, S. (2017). An evolutionary algorithm for optimizing the target ordering in ensemble of regressor chains. In *2017 IEEE congress on evolutionary computation (CEC)* (pp. 2015–2021). IEEE.
23. Santana, E.J., Mastelini, S.M., Barbon, S. Jr. (2017). Deep regressor stacking for air ticket prices prediction. In *Brazilian symposium of information systems* (pp. 216–233). SBSI 2017.
24. Sidike, P., Krieger, E., Alom, M.Z., Asari, V.K., Taha, T. (2017). A fast single-image super-resolution via directional edge-guided regularized extreme learning regression. In *Signal, image and video processing* (pp. 1–8).
25. Spyromitros-Xioufis, E., Tsoumakas, G., Groves, W., Vlahavas, I. (2016). Multi-target regression via input space expansion: treating targets as inputs. *Machine Learning*, *104*(1), 55–98.
26. Tsoumakas, G., Spyromitros-Xioufis, E., Vrekou, A., Vlahavas, I. (2014). Multi-target regression via random linear target combinations. In *Joint european conference on machine learning and knowledge discovery in databases* (pp. 225–240). Springer.
27. Wang, Q., Wu, Y., Shen, Y., Liu, Y., Lei, Y. (2015). Supervised sparse manifold regression for head pose estimation in 3d space. *Signal Processing*, *112*, 34–42.
28. Watanabe, S., Nakamura, A., Juang, B.H.F. (2014). Structural bayesian linear regression for hidden Markov models. *Journal of Signal Processing Systems*, *74*(3), 341–358.
29. Zhang, W., Liu, X., Ding, Y., Shi, D. (2012). Multi-output LS-SVR machine in extended feature space. In *CIMSA 2012 - 2012 IEEE Int. Conf. Comput. Int.ll. Meas. Syst. Appl. Proc.* (pp. 130–144). https://doi.org/10.1109/CIMSA.2012.6269600.

Saulo Martiello Mastelini received his BSc and MSc degrees in Computer Science in 2016 and 2018, respectively, at the State University of Londrina, Brazil. During his bachelor he researched in Computer Graphics, Computer Vision and Numerical methods for solving Partial Differential Equations. During his Master's he focused in Machine Learning techniques, with special attention for Multi-target regression tasks.

Victor Guilherme Turrisi da Costa BSc, is an MSc student in the Computer Science Department at State University of Londrina (UEL), Brazil. He received his BSc degree in Computer Science in 2017 at the State University of Londrina (UEL), Brazil. His research interests include Pattern Recognition, Machine Learning and botnet detection.

Everton Jose Santana is a Bsc student of Electrical Engineering at State University of Londrina, Brazil. From 2015 to 2016, he was an exchange student at Hanze University of Applied Sciences, the Netherlands, where he followed minors in Biomedical and Sensor System Engineering. His main research topics are Applied Mathematics, Instrumentation/Biomedical Engineering and Machine Learning.

Felipe Kenji Nakano received his BSc degree in Computer Science in 2016 at the State University of Londrina, Brazil. During his bachelor he worked with software development and researched Process Mining. Currently, he is finishing his MSc in Computer Science at Federal University of Sao Carlos. His topics of interest are Hierarchical Classification, Deep Learning and Active Learning.



Dr. Rodrigo Capobianco Guido received his BSc degrees in Computer Science and in Computer Engineering, his MSc degree in Electrical Engineering and his PhD degree in Computational Applied Physics, respectively from São Paulo State University (UNESP) at São José do Rio Preto - Brazil in 1998, from Educational Foundation at Votuporanga (FEV) - Brazil in 2003, from Campinas State University (UNICAMP) - Brazil in 2000 and from University of São Paulo (USP) at São Carlos - Brazil in 2003. From both BScs to PhD, all his titles focused on signal processing. In complement, he has already participated in two post-doctoral programs in signal processing at USP, from 2003 to 2007, and obtained the title of Associate Professor (Livre-Docência) in signal processing from the School of Engineering at São Carlos - Brazil (USP) in 2008. Dr. Guido has taught signal processing and electronics since 1999 and has published hundreds of scientific articles in IEEE and Elsevier journals, magazines, and conference proceedings. He is serving, or recently served, as an area-editor, as an associate-editor, and as a guest-editor for respected scientific journals, such as IEEE Signal Processing Magazine, IEEE Transactions on Audio, Speech and Language Processing, Elsevier Pattern Recognition Letters, Elsevier Neurocomputing, Elsevier Computers in Biology and Medicine, just to mention a few. Complementarily, he has served as an organiser and chairman for many IEEE conferences along the years. Dr. Guido has received several grants and awards from Brazilian agencies, especially from The State of São Paulo Research Foundation (FAPESP) and from National Council of Research and Development (CNpQ). He has also supervised many theses in his field and is a senior member of the IEEE. Currently, he is an associate professor at UNESP in São José do Rio Preto - Brazil. Dr. Guido has focused his research activities on digital signal processing (DSP) with particular focus on speech processing, i.e., speech transmission and reception, speech enhancement, speech recognition, speaker identification and verification, speech analysis for biomedical purposes, voice morphing, speech emotion classification, voice activity detection, speech synthesis, and so on, specially based on wavelets associated with algorithms for machine learning.

Ricardo Cerri obtained his Bachelor in Computer Science from São Paulo State University (UNESP/Brazil), and his MSc and PhD in Computer Science and Computational Mathematics from University of São Paulo (ICMC/USP/Brazil). He has experience working mainly with Bioinformatics and Machine Learning, focusing on advanced methods for data classification, such as multi-output and structured learning. Currently he holds the position of Assistant Professor at the Department of Computer Science from Federal University of São Carlos (UFSCar/Brazil).

Sylvio Barbon Jr. PhD, is Associate Professor and leader of the research group that studies machine learning in the Computer Science Department at State University of Londrina (UEL), Brazil. He received his BSc degree in Computer Science in 2005, MSc degree in Computational Physics from University of São Paulo (2007), degree in Computational Engineering during 2008 and PhD degree (2011) from IFSC/USP such as the MSc degree. During 2017, he was visiting researcher at University of Modena and Reggio Emilia (Italy) working on multispectral analysis and machine learning. Still in 2017, as visiting researcher at Universita Degli Studi Di Milano (Italy) focused on Stream and Process Mining. He is currently a professor in postgraduate and graduate programs. His research interests include Digital Signal Processing, Pattern Recognition and Machine Learning.