Mach Learn (2016) 104:55–98
DOI 10.1007/s10994-016-5546-z

Multi-target regression via input space expansion: treating targets as inputs

Eleftherios Spyromitros-Xioufis[1] ·
Grigorios Tsoumakas[1] · **William Groves**[2] · **Ioannis Vlahavas**[1]

Received: 15 July 2015 / Accepted: 16 January 2016 / Published online: 19 February 2016

Abstract In many practical applications of supervised learning the task involves the prediction of multiple target variables from a common set of input variables. When the prediction targets are binary the task is called multi-label classification, while when the targets are continuous the task is called multi-target regression. In both tasks, target variables often exhibit statistical dependencies and exploiting them in order to improve predictive accuracy is a core challenge. A family of multi-label classification methods address this challenge by building a separate model for each target on an expanded input space where other targets are treated as additional input variables. Despite the success of these methods in the multi-label classification domain, their applicability and effectiveness in multi-target regression has not been studied until now. In this paper, we introduce two new methods for multi-target regression, called *stacked single-target* and *ensemble of regressor chains*, by adapting two popular multi-label classification methods of this family. Furthermore, we highlight an inherent problem of these methods—a discrepancy of the values of the additional input variables between training and prediction—and develop extensions that use out-of-sample estimates of the target variables during training in order to tackle this problem. The results of an extensive experimental evaluation carried out on a large and diverse collection of datasets show that, when the discrepancy is appropriately mitigated, the proposed methods attain consistent improve-

Editor: Johannes Fürnkranz.

✉ Eleftherios Spyromitros-Xioufis
espyromi@csd.auth.gr

Grigorios Tsoumakas
greg@csd.auth.gr

William Groves
groves@cs.umn.edu

Ioannis Vlahavas
vlahavas@csd.auth.gr

1 Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece

2 Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, USA

ments over the independent regressions baseline. Moreover, two versions of Ensemble of Regression Chains perform significantly better than four state-of-the-art methods including regularization-based multi-task learning methods and a multi-objective random forest approach.

Keywords Multi-target regression · Multi-label classification · Stacking · Chaining

1 Introduction

Multi-target regression (MTR), also known as multivariate or multi-output regression, refers to the task of predicting multiple continuous variables using a common set of input variables. Such problems arise in various fields including ecological modeling (Kocev et al. 2009; Dzeroski et al. 2000) (e.g. predicting the abundance of plant species using water quality measurements), economics (Ghosn and Bengio 1996) (e.g. predicting stock prices from econometric variables) and energy (e.g. predicting energy production in solar/wind farms using historical measurements and weather forecast information). Given the importance and diversity of its applications, it is not surprising that research on this topic has started as early as 40 years ago in Statistics (Izenman 1975).

Recently, a closely related task called multi-label classification (MLC) (Tsoumakas et al. 2010; Zhang and Zhou 2014) has received increased attention by Machine Learning researchers. Similarly to MTR, MLC deals with the prediction of multiple variables using a common set of input variables. However, prediction targets in MLC are binary. In fact, the two tasks can be thought of as instances of the more general learning task of multi-target prediction where targets can be continuous, binary, ordinal, categorical or even of mixed type. The baseline approach of learning a separate model for each target applies to both MTR and MLC. Moreover, they share the same core challenge of exploiting dependencies between targets (in addition to dependencies between targets and inputs) in order to improve prediction accuracy, as acknowledged by researchers working in both tasks (e.g. Izenman 2008; Dembczynski et al. 2012). Despite their commonalities, MTR and MLC have typically been treated in isolation and only few works (Blockeel et al. 1998; Weston et al. 2002; Teh et al. 2005; Balasubramanian and Lebanon 2012) have given a general formulation of their key ideas, recognizing the dual applicability of their approaches.

Motivated by the tight connection between the two tasks, this paper looks at a family of MLC methods that, despite being almost directly applicable to MTR problems, have not been applied so far in this domain. In particular, we consider methods that decompose the MLC task into a series of binary classification tasks, one for each label. This category, includes the typical one-versus-all or *Binary Relevance* approach that assumes label independence but also approaches that model label dependencies by building models that treat other labels as additional input variables (meta-inputs). In this work we adapt two popular methods of this kind (Godbole and Sarawagi 2004; Read et al. 2011) for MTR, contributing two new MTR methods: *Stacked single-target* (SST) and *Ensemble of Regressor Chains* (ERC). Both methods have been very successful in the MLC domain and provided inspiration for many subsequent works (Cheng and Hüllermeier 2009; Dembczynski et al. 2010; Kumar et al. 2012; Read et al. 2014).

Although the adaptation is trivial (as it basically consists of employing a regression instead of a binary classification algorithm to solve each single-target prediction task), it widens the applicability of existing approaches and increases our understanding of challenges shared by both learning tasks, such as the modeling of target dependencies. This kind of abstraction of key ideas from solutions tailored to related problems can sometimes offer additional advantages, such as improving the modularity and conceptual simplicity of learning techniques and avoiding reinvention of the same solutions.[1]

In addition to evaluating the direct adaptations of the corresponding MLC methods in the MTR domain, we also take a careful look at the treatment of targets as additional input variables and spot a shortcoming that was overlooked in the original MLC formulations of both methods. Specifically, we notice that in both methods the values of the meta-inputs are generated differently between training and prediction, causing a discrepancy that is shown to drastically downgrade their performance. To tackle this problem, we develop extended versions of the two methods that manage to decrease the discrepancy by using out-of-sample estimates of the targets during training. These estimates are obtained via an internal cross-validation methodology.

The performance of the proposed methods is comprehensively analyzed based on a large experimental study that includes 18 diverse real-world datasets, 14 of which are firstly used in this paper and are made publicly available for future benchmarks. The experimental results reveal that, affected by the discrepancy problem, the direct adaptations of the corresponding MLC methods fail to obtain better accuracy than the baseline approach that performs independent regressions. On the other hand, the extended versions obtain consistent improvements against the baseline, confirming the effectiveness of the proposed solution. Furthermore, extended versions of ERC obtain significantly better accuracy than state-of-the-art methods, including a method based on ensembles of multi-objective decision trees (Kocev et al. 2007) and a recent regularization-based multi-task learning method (Jalali et al. 2010, 2013). Moreover, it is shown that, compared to the rest of the methods, the extended versions of ERC are associated with the smallest risk of decreasing the accuracy of the baseline, an appealing property.

The rest of the paper is organized as follows: Sect. 2 presents the SST and ERC methods and describes the discrepancy problem and the proposed solution. Section 3 discusses related work from the MTR field, including well-known statistical procedures and multi-task learning methods, and points out differences with previous work on the discrepancy problem. The details of the experimental setup (method configuration, evaluation methodology, datasets) are given in Sects. 4 and 5 presents and discusses the experimental results. Finally, Sect. 6 offers our conclusion and outlines future work directions.

2 Methods

We first formally describe the MTR task and provide the notation that will be used subsequently for the description of the methods. Let $\mathbf{X}$ and $\mathbf{Y}$ be two random vectors where $\mathbf{X}$ consists of d input variables $X_1, \ldots, X_d$ and $\mathbf{Y}$ consists of m target variables $Y_1, \ldots, Y_m$. We assume that samples of the form $(\mathbf{x}, \mathbf{y})$ are generated i.i.d. by some source according to a joint probability distribution $\mathbf{P}(\mathbf{X}, \mathbf{Y})$ on $\mathscr{X} \times \mathscr{Y}$ where $\mathscr{X} = R^d$[2] and $\mathscr{Y} = R^m$ are the domains of $\mathbf{X}$ and $\mathbf{Y}$ and are often referred to as the input and the output space. In a sample $(\mathbf{x}, \mathbf{y})$, $\mathbf{x} = [x_1, \ldots, x_d]$ is the input vector and $\mathbf{y} = [y_1, \ldots, y_m]$ is the output vector which are realizations of $\mathbf{X}$ and $\mathbf{Y}$ respectively. Given a set $D = \{(\mathbf{x}^1, \mathbf{y}^1), \ldots, (\mathbf{x}^n, \mathbf{y}^n)\}$ of n training examples, the goal in MTR is to learn a model $\mathbf{h} : \mathscr{X} \rightarrow \mathscr{Y}$ that given an input

[1] See NIPS'11 workshop on relations among machine learning problems at http://rml.anu.edu.au/.

[2] $\mathscr{X} = R^d$ is used only for the sake of brevity. The domain of the input variables can also be discrete.

vector $\mathbf{x}$, is able to predict an output vector $\hat{\mathbf{y}} = \mathbf{h}(\mathbf{x})$ that best approximates the true output vector $\mathbf{y}$.

In the baseline Single-Target (ST) method, a multi-target model $\mathbf{h}$ is comprised of m single-target models $h_j : \mathscr{X} \rightarrow R$ where each model h_j is trained on a transformed training set $D_j = \{(\mathbf{x}^1, y_j^1), \ldots, (\mathbf{x}^n, y_j^n)\}$ to predict the value of a single target variable Y_j. This way, target variables are modeled independently and no attempt is made to exploit potential dependencies between them. Despite the simplicity of the ST approach, several empirical studies (e.g. Luaces et al. 2012) have shown that Binary Relevance, its MLC counterpart, often obtains comparable performance with more sophisticated MLC methods that model label dependencies, especially in cases where the underlying single-target prediction model is well fitted to the data (Dembczynski et al. 2012; Read and Hollmén 2014, 2015). A theoretical explanation of these results was offered by Dembczynski et al. (2012) who showed that modeling the marginal conditional distributions $P(Y_i|\mathbf{x})$ of the labels (as done by Binary Relevance) can be sufficient for getting good results in multi-label losses whose risk minimizers can be expressed in terms of marginal distributions (e.g. Hamming loss).

2.1 Stacked single-target

Stacked single-target (SST) is inspired from the Stacked Binary Relevance method (Godbole and Sarawagi 2004) where the idea of stacked generalization (Wolpert 1992) was applied in a MLC context. The training of SST consists of two stages. In the first stage, m independent single-target models $h_j : \mathscr{X} \rightarrow R$ are learned as in ST. However, instead of directly using these models for prediction, SST involves an additional training stage where a second set of m meta models $h'_j : \mathscr{X} \times R^m \rightarrow R$ are learned, one for each target Y_j. Each meta model h'_j is learned on a transformed training set $D'_j = \{(\mathbf{x}'^1, y_j^1), \ldots, (\mathbf{x}'^n, y_j^n)\}$, where the original input vectors of the training examples $(\mathbf{x}^i)$ have been augmented by estimates of the values of their target variables $(\hat{y}_1^i, \ldots, \hat{y}_m^i)$ to form expanded input vectors $\mathbf{x}'^i = [\mathbf{x}^i, \hat{y}_1^i, \ldots, \hat{y}_m^i]$. These estimates are obtained by applying the first stage models to the examples of the training set.

To obtain predictions for an unknown instance $\mathbf{x}^q$, the first stage models are first applied and an output vector $\hat{\mathbf{y}}^q = [h_1(\mathbf{x}^q), \ldots, h_m(\mathbf{x}^q)]$ is obtained. Then, the second stage models are applied on transformed input vectors $\mathbf{x}'^q = [\mathbf{x}^q, \hat{\mathbf{y}}^q]$ to produce the final output vector $\tilde{\mathbf{y}}^q = [h'_1(\mathbf{x}_1'^q), \ldots, h'_m(\mathbf{x}_m'^q)]$. The training and prediction procedures of SST are graphically illustrated in Fig. 1.

2.2 Ensemble of regressor chains

Regressor Chains (RC) is derived from Classifier Chains (Read et al. 2011), a recently proposed MLC method based on the idea of chaining binary models. The training of RC consists of selecting a random chain (permutation) of the set of target variables and then building a separate regression model for each target. Assuming that the chain $C = \{Y_1, Y_2, \ldots, Y_m\}$ (C represents an ordered set) is selected, the first model concerns the prediction of Y_1, has the form $h_1 : \mathscr{X} \rightarrow R$ and is the same as the model built by the ST method for this target. The difference in RC is that subsequent models h_j, $j > 1$ are learned on transformed training sets $D'_j = \{(\mathbf{x}_j'^1, y_j^1), \ldots, (\mathbf{x}_j'^n, y_j^n)\}$, where the original input vectors of the training examples have been augmented by the actual values of all previous targets of the chain to form expanded input vectors $\mathbf{x}_j'^i = [x_1^i, \ldots, x_d^i, y_1^i, \ldots, y_{j-1}^i]$. Thus, the models built for targets Y_j have the form $h_j : \mathscr{X} \times R^{j-1} \rightarrow R$.

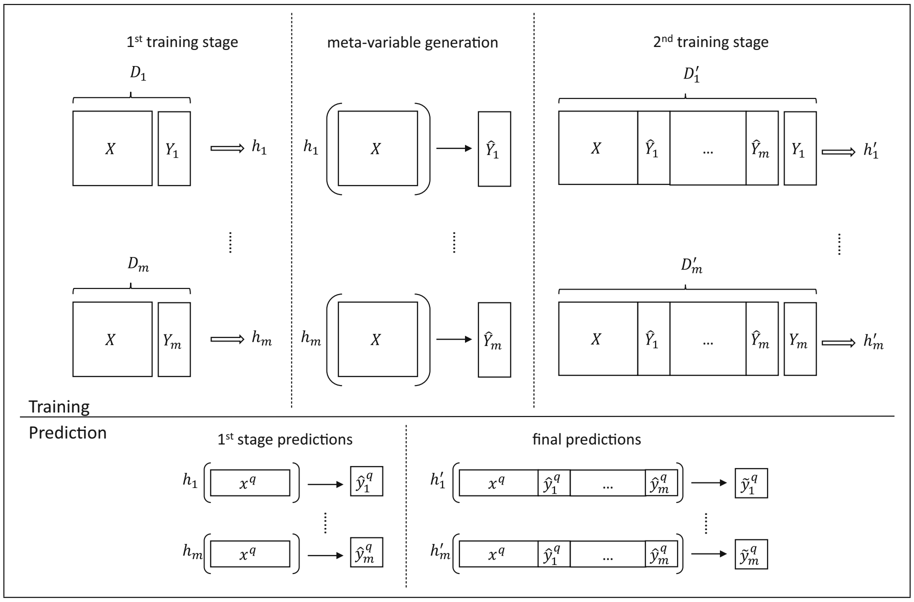


Fig. 1 Graphical illustration of SST's training and prediction procedures

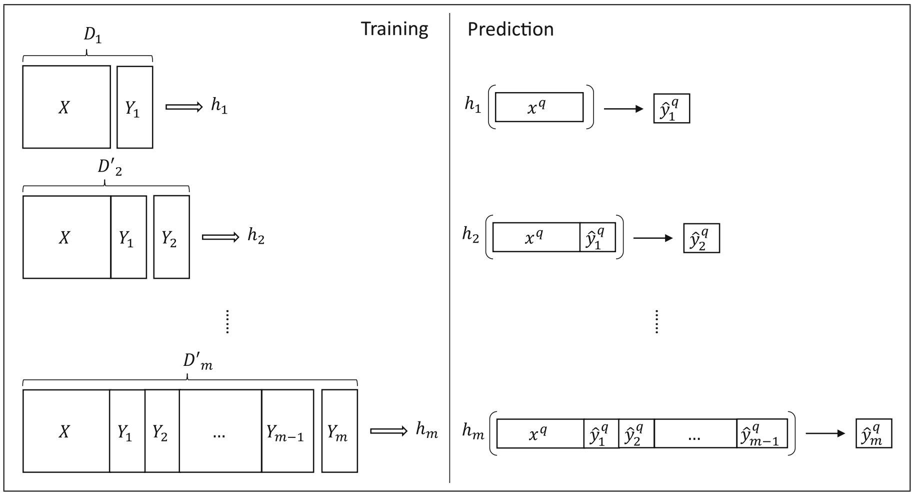


Fig. 2 Graphical illustration of RC's training and prediction procedures

Given such a chain of models, the output vector $\hat{\mathbf{y}}^q$ of an unknown instance $\mathbf{x}^q$ is obtained by sequentially applying the models h_j, thus $\hat{\mathbf{y}}^q = [h_1(\mathbf{x}^q), h_2(\mathbf{x}_2'^q), \ldots, h_m(\mathbf{x}_m'^q)]$ where $\mathbf{x}_j'^q = [x_1^q, \ldots, x_d^q, \hat{y}_1^q, \ldots, \hat{y}_{j-1}^q]$. Note that since the true values $y_1^q, \ldots, y_{j-1}^q$ of the target variables are not available at prediction time, the method relies on estimates of these values obtained by applying the models $h_1, \ldots, h_{j-1}$. The training and prediction procedures of RC are graphically illustrated in Fig. 2.

One notable property of RC is that it is sensitive in the selected chain ordering. To alleviate this issue, Read et al. (2011) proposed an ensemble scheme called Ensemble of Classifier Chains where a set of k Classifier Chains models with different random chains are built on bootstrap samples of the training set and the final predictions come from majority voting. This scheme has been shown to consistently improve the accuracy of a single Classifier Chain in the classification domain. We apply the same idea on RC and compute the final predictions by taking the mean of the k estimates for each target. The resulting method is called *Ensemble of Regressor Chains* (ERC).

2.3 Theoretical insights into stacking and chaining

Both stacking and chaining have enjoyed significant attention from the MLC community, mainly due to their high performance and conceptual simplicity. A number of recent works have attempted a theoretical analysis of the methods (Dembczynski et al. 2010, 2012; Read and Hollmén 2015). Adopting a statistical perspective, the authors of Dembczynski et al. (2010, 2012) distinguish between two types of label dependence:

- unconditional, where $P(\mathbf{Y}) \neq \prod_{i=1}^{m} P(Y_i)$; and
- conditional, where $P(\mathbf{Y}|\mathbf{x}) \neq \prod_{i=1}^{m} P(Y_i|\mathbf{x})$,

and show that modeling them is important for improving generalization performance. According to this analysis, stacking is interpreted as a method that models unconditional label dependence and is more suitable for minimizing label-wise decomposable multi-label loss functions,[3] while chaining is interpreted as a method that models conditional dependence and is more suitable for minimizing multi-label loss functions that cannot be decomposed label-wise.

Another interesting interpretation is offered by Read and Hollmén (2015) who show that Binary Relevance can (under certain conditions) achieve optimal performance in any dataset, and that improvements over the independent approach are often the result of using an inadequate base learner. Under this view, stacking and chaining can be considered as 'deep' independent learners who owe their improved performance over Binary Relevance (when the same base learner is used) to the use of labels as nodes in the inner layers of a deep neural network. These nodes represent readily available[4] (in the training phase), high-level transformations of the original inputs. This interpretation of stacking and chaining applies directly to the MTR versions of these methods that we present here.

From a bias-variance perspective, we observe that by introducing additional features to single-target models, SST and ERC have the effect of decreasing their bias at the expense of an increased variance. This suggests that whenever the increase in variance is outweighed by the decrease in bias, one should expect gains in generalization performance over ST. This also hints that both methods will probably benefit from being combined with a base regressor that includes a variance reduction mechanism like bagged (Breiman 1996) regression trees.[5] As shown in Munson and Caruana (2009), bagged trees not only ignore irrelevant features but can also exploit features that contain useful but noisy information. Both properties are very important in the context of SST and ERC because some

[3] Note, however, that this analysis concerns a version of stacking that does not include the original input variables in the input space of the second stage models.

[4] This is in contrast with traditional deep learning where high-level feature representations are typically learned from the data in an unsupervised way.

[5] An explicit feature selection could alternatively be applied as a means of variance reduction.

of the extra features that they introduce might be irrelevant (e.g. whenever two target variables are statistically independent) and/or noisy (as discussed in the following subsection).

2.4 Generation of meta-inputs

Both SST and ERC are based on the same core idea of treating other prediction targets as additional input variables. These meta-inputs differ from ordinary inputs in the sense that while their actual values are available at training time, they are missing during prediction. Thus, during prediction both methods have to rely on estimates of these values which come either from ST (in the case of SST) or from RC (in the case of ERC) models built on the training set. An important question that is answered differently by each method is the following: *What type of values should be used at training time for the meta-inputs?* SST uses estimates of the variables obtained by applying the first stage models on the training examples, while ERC uses their actual values. We observe that in both cases a core assumption of supervised learning is violated: that the training and testing data should be identically and independently distributed. In the SST case, the in-sample estimates that are used to form the training examples of the second stage models will typically be more accurate than the out-of-sample estimates used at prediction time. The situation is even more problematic in the case of ERC since the actual target values are used during training. In both cases, some of the input variables that are used by the underlying regression algorithm during model induction, become noisy (or noisier in the case of SST) at prediction time and, as a result, the induced model might wrongly estimate (overestimate) their usefulness.

To mitigate this problem, we propose the use of out-of-sample estimates of the targets during training in order to increase the compatibility between the training values of the target variables and the values used during prediction. One way to obtain such estimates is to use a subset of the training set for building the first stage ST models (in the case of SST) or the RC models (in the case of ERC) and apply them to the held-out part. However, this approach would lead to reduced second stage training sets for SST as only the examples of the held-out set would be available for training the second stage models. The same holds for ERC where the chained RC models would be trained on training sets of decreasing size. The solution that we propose to this problem is the use of an internal f-fold cross-validation approach that allows obtaining out-of-sample estimates of the target variables for all the training examples. Compared to the actual target values or the in-sample estimates of the targets, the cross-validation estimates are expected to better resemble the values that are used during prediction. As a result, we expect that the contribution of the meta-inputs to the prediction of each target will be better estimated by the underlying regression algorithm.

The training procedures of the extended SST (denoted as SST_{cv}) and RC (denoted as RC_{cv}) methods are outlined in Algorithms 1 and 3. ERC_{cv} consists of simply repeating the RC_{cv} procedure k times with different random chains. The corresponding prediction procedures are presented in Algorithms 2 and 4. Note that the prediction procedures of the original and the extended versions of each method coincide. In Sect. 5 we compare the performance of the extended versions of SST and ERC with the performance of the directly adapted variants, henceforth denoted as SST_{train} and ERC_{true}. To better study the effects of the discrepancy problem, the comparison also includes SST using the actual target values (SST_{true}) and ERC using in-sample estimates of the target variables (ERC_{train}).

Algorithm 1: SST_{cv} training

```
Input: Training set $D$, number of internal cross-validation folds $f$
Output: 1st & 2nd stage models $h_j$ & $h'_j$, $j = 1 \dots m$
// Build 1st stage models
for $j = 1$ to $m$ do
    $D_j = \{(\mathbf{x}^1, y_j^1), \dots, (\mathbf{x}^n, y_j^n)\}$                // transform $D$ to $D_j$
    $h_j : D_j \rightarrow R$                                  // build model for $Y_j$ using $D_j$
    split $D_j$ randomly into $f$ disjoint parts $D_j^i$, $i = 1 \dots f$
    for $i = 1$ to $f$ do
        $h_j^i : D_j \setminus D_j^i \rightarrow R$                  // build model for $Y_j$ using $D_j \setminus D_j^i$
// Generate 2nd stage training sets
for $j = 1$ to $m$ do
    $D'_j \leftarrow \emptyset$
    for $i = 1$ to $f$ do
        $D'^i_j \leftarrow \emptyset$
        foreach $\mathbf{x}^k \in D_j^i$ do
            $\hat{\mathbf{y}}^k = [h_1^i(\mathbf{x}^k), \dots, h_m^i(\mathbf{x}^q)]$
            $\mathbf{x}'^k = [\mathbf{x}^k, \hat{\mathbf{y}}^k]$                     // concatenate $\mathbf{x}^k$ and $\hat{\mathbf{y}}^k$
            $D'^i_j = D'^i_j \cup (\mathbf{x}'^k, y_j^k)$
        $D'_j = D'_j \cup D'^i_j$
// Build 2nd stage models
for $j = 1$ to $m$ do
    $h'_j : D'_j \rightarrow R$
```

Algorithm 2: SST prediction

```
Input: Unknown instance $x^q$, 1st & 2nd stage models $h_j$ & $h'_j$, $j = 1 \dots m$
Output: Output vector $\tilde{\mathbf{y}}^q$
$\hat{\mathbf{y}}^q = \tilde{\mathbf{y}}^q = \mathbf{0}$
// Apply the 1st stage models
for $j = 1$ to $m$ do
    $\hat{y}_j^q = h_j(\mathbf{x}^q)$
$\mathbf{x}'^q = [\mathbf{x}^q, \hat{\mathbf{y}}^q]$                              // concatenate $\mathbf{x}^q$ and $\hat{\mathbf{y}}^q$
// Apply the 2nd stage models
for $j = 1$ to $m$ do
    $\tilde{\mathbf{y}}_j^q = h'_j(\mathbf{x}'^q)$
```

2.5 Discussion

Besides the type of values that each method uses for the meta-inputs at training time, SST and ERC have additional conceptual differences. A notable one is that the model built for each target Y_j by SST, uses all other targets as inputs while in RC each model involves only targets that precede Y_j in a random chain. As a result, the model built for Y_j by RC, cannot benefit from statistical relationships with targets that appear later than Y_j in the chain. This potential disadvantage of RC is partially overcome by ERC since each target is included in multiple random chains and, therefore, the probability that other targets will

Algorithm 3: RC_{cv} training

```
Input: Training set $D$, number of internal cross-validation folds $f$
Output: Chained models $h_j$, $j = 1 \dots m$
// Generate $D'_1$
$D'_1 = \{(\mathbf{x}^1, y^1_1), \dots, (\mathbf{x}^n, y^n_1)\}$            // transform $D$ to $D'_1$
for $j = 1$ to $m$ do
    $h_j : D'_j \rightarrow R$                       // build model for $Y_j$ using $D'_j$
    if $j < m$ then
        // Generate $D'_{j+1}$
        $D'_{j+1} \leftarrow \emptyset$
        split $D'_j$ randomly into $f$ disjoint parts $D'^i_j$, $i = 1 \dots f$
        for $i = 1$ to $f$ do
            $h^i_j : D'_j \setminus D'^i_j \rightarrow R$         // build model for $Y_j$ using $D'_j \setminus D'^i_j$
            foreach $\mathbf{x}'^k_j \in D'^i_j$ do
                $x'^i_{j+1} = x'^i_j$
                $\hat{y}^k_j = h^i_j(\mathbf{x}'^k_j)$
                $\mathbf{x}'^k_{j+1} = [\mathbf{x}'^k_j, \hat{y}^k_j]$            // append $\mathbf{x}'^k_j$ with $\hat{y}^k_j$
                $D'_{j+1} = D'_{j+1} \cup (\mathbf{x}'^k_{j+1}, y^k_{j+1})$
```

Algorithm 4: RC prediction

```
Input: Unknown instance $x^q$, chain models $h_j$, $j = 1 \dots m$
Output: Output vector $\hat{\mathbf{y}}^q$
$\hat{\mathbf{y}}^q = \mathbf{0}$
$\mathbf{x}'^q_1 = \mathbf{x}^q$
for $j = 1 \dots m$ do
    $\hat{y}^q_j = h_j(\mathbf{x}'^q_j)$
    if $j < m$ then
        $\mathbf{x}'^q_{j+1} = [\mathbf{x}'^q_j, \hat{y}^q_j]$            // append $\mathbf{x}'^q_j$ with $\hat{y}^q_j$
```

precede it is increased. At a first glance, SST seems to represent a more straightforward way of including all the available information about other targets. However, we should take into account that, since both methods rely on estimates of the meta-inputs at prediction time (as discussed in previous subsection), the more the meta-inputs that are included in the input space, the higher the amount of error accumulation that is risked at prediction time. From this perspective, ERC seems to adopt a more cautious approach than SST. On the other hand, the estimates of the meta-inputs that are used by the second stage models in SST come from independent models, while the estimates of the meta-inputs used by each model in RC (and ERC) come from models that include information about other targets and thus involve a higher risk of becoming noisy. Overall, there seems to be a trade-off between using the additional information available in the targets and the noise that this information comes with. Which of the two methods (and which variant) achieves a better balance in this trade-off is revealed by the experimental analysis in Sect. 5.

2.6 Complexity analysis

In this section we discuss the time complexity of all variants of the proposed methods at training and test time, given a single-target regression algorithm with training complexity $O(g_{tr}(n, d))$ and test complexity $O(g_{te}(n, d))$ for a dataset with n examples and d input variables. The training and test complexities of the ST method are $O(m \cdot g_{tr}(n, d))$ and $O(m \cdot g_{te}(n, d))$ respectively, as it involves training and querying m independent single-target models.

With respect to SST, the method builds $2 \cdot m$ models at training time, all of which are queried at prediction time. In all variants of the method, half of the models are built on the original input space and half of the models are built on an input space augmented by m meta-inputs. Thus, in the case of SST$_{true}$, where the meta-inputs are readily available, the training and test complexities are $O(m \cdot (g_{tr}(n, d) + g_{tr}(n, d+m)))$ and $O(m \cdot (g_{te}(n, d) + g_{te}(n, d+m)))$ respectively. Given that in most cases (see Table 3) the number of targets is much smaller than the number of inputs, i.e. $m \ll d$, the effective training and test complexities of SST$_{true}$ become $O(m \cdot g_{tr}(n, d))$ and $O(m \cdot g_{te}(n, d))$ respectively, thus same with ST's complexities. SST$_{train}$ and SST$_{cv}$ have the same test complexity with SST$_{true}$ but a larger training complexity because of the process of generating estimates for the meta-inputs. In the SST$_{train}$ case, the training complexity is $O(m \cdot g_{tr}(n, d) + m \cdot g_{te}(n, d))$ because the m first-stage models are applied to obtain estimates for all the training examples. For most regression algorithms (e.g. regression trees), the computational cost of making predictions for n instances is much smaller than the cost of training on n examples. For instance, the training complexity of a typical binary regression tree learner is $O(n \cdot d^2)$ (Su and Zhang 2006) while the test complexity is $O(n \cdot \log_2 d)$. Thus, practically, the training complexity of SST$_{train}$ is similar to that of SST$_{true}$. When it comes to SST$_{cv}$, in addition to the m first-stage models, f additional models are built on $\frac{f-1}{f} \cdot n$ examples each. Therefore, the training complexity of SST$_{cv}$ is $O(m \cdot g_{tr}(n, d) + m \cdot f \cdot g_{tr}(\frac{f-1}{f} \cdot n, d) + m \cdot g_{te}(n, d)) \approx O(f \cdot m \cdot g_{tr}(n, d) + m \cdot g_{te}(n, d))$. Given

that $g_{te}(n, d) \ll g_{tr}(n, d)$, we conclude that the training complexity of SST$_{cv}$ is roughly f times ST's training complexity. Also, note that SST$_{train}$ and SST$_{cv}$ can be parellelized stage-wise both at training and at prediction time, i.e. all single-target models within the same level can be trained and queried independently, while SST$_{true}$ is fully parallelizable at training time (all single-target models can be trained independently) and stage-wise parallelizable at test time.

In ERC, each RC model consists of a chain of m models built on input spaces augmented by $\{0, 1, \ldots, m-1\}$ meta-inputs, thus $\frac{m-1}{2}$ meta-inputs on average. In the case of ERC$_{true}$ and for an ensemble size of k RC models, the training and test complexities are $O(k \cdot m \cdot g_{tr}(n, d + (\frac{m-1}{2}))$ and $O(k \cdot m \cdot g_{te}(n, d + (\frac{m-1}{2}))$ respectively. Given, as before, that $m \ll d$, the training complexity of ERC$_{true}$ becomes $O(k \cdot m \cdot g_{tr}(n, d))$ and its test complexity becomes $O(k \cdot m \cdot g_{te}(n, d))$, thus k times ST's complexity in both cases. Following a similar reasoning as we did above for SST, we can show that the training complexity of ERC$_{train}$ is similar to that of ERC$_{true}$ and that the training complexity of ERC$_{cv}$ is $O(k \cdot f \cdot m \cdot g_{tr}(n, d))$, i.e. $k \cdot f$ times ST's training complexity. Obviously, the test complexities of both ERC$_{train}$ and ERC$_{cv}$ are the same as ERC$_{true}$'s test complexity. With respect to parellelization, we observe that each member of an ERC$_{train}$ or ERC$_{cv}$ ensemble can be trained independently, while ERC$_{true}$ is fully parallelizable at training time, i.e. all $k \cdot m$ single-target models can be trained independently. For all ERC variants, test time parallelization is also possible since each ensemble member can be queried independently.

Table 1 Training and test complexities of the proposed methods with single- and multi-core implementations

Method		Training complexity		Test complexity	
		Single-core	Multi-core	Single-core	Multi-core
SST	*true*	$O(m \cdot g_{tr}(n, d))$	$O(g_{tr}(n, d))$	$O(m \cdot g_{te}(n, d))$	$O(g_{te}(n, d))$
	train	$O(m \cdot g_{tr}(n, d))$	$O(g_{tr}(n, d))$	$O(m \cdot g_{te}(n, d))$	$O(g_{te}(n, d))$
	cv	$O(f \cdot m \cdot g_{tr}(n, d))$	$O(g_{tr}(n, d))$	$O(m \cdot g_{te}(n, d))$	$O(g_{te}(n, d))$
ERC	*true*	$O(k \cdot m \cdot g_{tr}(n, d))$	$O(g_{tr}(n, d))$	$O(k \cdot m \cdot g_{te}(n, d))$	$O(m \cdot g_{te}(n, d))$
	train	$O(k \cdot m \cdot g_{tr}(n, d))$	$O(m \cdot g_{tr}(n, d))$	$O(k \cdot m \cdot g_{te}(n, d))$	$O(m \cdot g_{te}(n, d))$
	cv	$O(k \cdot f \cdot m \cdot g_{tr}(n, d))$	$O(m \cdot g_{tr}(n, d))$	$O(k \cdot m \cdot g_{te}(n, d))$	$O(m \cdot g_{te}(n, d))$

n, d and m denote the numbers of data points, inputs, and targets respectively. k denotes the number of chains in ERC and f the number of internal cross-validation folds in the cv variants of SST and ERC

Table 1 summarizes the training and test complexities of each method assuming a single-core implementation as well as the minimum possible complexity when a multi-core implementation is used. Note that, as shown in the table, SST$_{cv}$ and ERC$_{cv}$ have the same multi-core complexity with SST$_{train}$ and ERC$_{train}$ respectively because their internal cross-validation procedure can also be parallelized.

3 Related work

3.1 Multi-target regression

MTR was first studied in Statistics under the term multivariate regression with Reduced Rank Regression (RRR) (Izenman 1975), FICYREG (Merwe and Zidek 1980) and two-block PLS (Wold 1985) (the multiple response version of PLS) being three of the earliest methods. Among these methods, two-block PLS has been used more widely, especially in Chemometrics. More recently, the Curds and Whey (C&W) method was proposed (Breiman and Friedman 1997) and was found to outperform RRR, FICYREG and two-block PLS. As noted by Breiman and Friedman (1997), C&W, RRR and FICYREG can all be expressed using the same generic form $\tilde{\mathbf{y}} = \mathbf{B}\hat{\mathbf{y}}$, where $\hat{\mathbf{y}}$ are estimates obtained by applying ordinary least squares regression on the target variables and $\mathbf{B}$ is a matrix that modifies these estimates in order to obtain a more accurate prediction $\tilde{\mathbf{y}}$, under the assumption that the targets are correlated.

In all methods, $\mathbf{B}$ can be expressed as $\mathbf{B} = \hat{\mathbf{T}}^{-1}\mathbf{D}\hat{\mathbf{T}}$, where $\hat{\mathbf{T}}$ is the matrix of sample canonical co-ordinates and $\mathbf{D}$ is a diagonal "shrinking" matrix that is obtained differently in each method. SST is highly similar to these methods but allows a more general formulation of the MTR problem. Firstly, SST does not impose any restriction to the family of models that generate the uncorrected (first stage) estimates in contrast to these approaches that use estimates obtained from least squares regression. Secondly, the correction of the estimates applied by SST comes from a learning procedure that jointly considers target and input variables rather than target variables alone.

As shown by Breiman and Friedman (1997), the above methods can be described by an alternative but equivalent scheme. According to this, $\mathbf{y}$ is first transformed to the canonical co-ordinate system $\mathbf{y}' = \hat{\mathbf{T}}\mathbf{y}$, then separate least squares regression is performed on each $\mathbf{y}'$

to obtain $\hat{\mathbf{y}}'$, these estimates are scaled by $\mathbf{D}$ to obtain $\tilde{\mathbf{y}}' = \mathbf{D}\hat{\mathbf{y}}'$ and finally transformed back to the original output space $\tilde{\mathbf{y}} = \hat{\mathbf{T}}^{-1}\tilde{\mathbf{y}}'$. As discussed by Dembczynski et al. (2012), from this perspective, these methods fall under a more general scheme where the output space is first transformed, single-target regressors are then trained on the transformed output space and an inverse transformation is performed (possibly along with shrinkage/regularization) to obtain predictions for the original targets. Due to its generality, this scheme has been adopted by a number of recent methods in both MLC (Hsu 2009; Zhang and Schneider 2011, 2012; Tai and Lin 2012) and MTR (Balasubramanian and Lebanon 2012; Tsoumakas et al. 2014).

A large number of MTR methods are derived from the predictive clustering tree (PCT) framework (Blockeel et al. 1998). The main difference between the PCT algorithm and a standard decision tree is that the variance and the prototype functions are treated as parameters that can be instantiated to fit the given learning task. Such an instantiation for MTR tasks are the multi-objective decision trees (MODTs) where the variance function is computed as the sum of the variances of the targets, and the prototype function is the vector mean of the target vectors of the training examples falling in each leaf (Blockeel et al. 1998, 1999). Bagging and random forest ensembles of MODTs were developed by Kocev et al. (2007) and were found significantly more accurate than MODTs and equally good or better than ensembles of single-objective decision trees for both regression and classification tasks. In particular, multi-objective random forests yielded better performance than multi-objective bagging.

Methods that deal with the prediction of multiple target variables can be found in the literature of the related learning task of multi-task learning. According to Caruana (1997), multi-task learning is a form of inductive transfer (Pratt 1992) where the aim is to improve generalization accuracy on a set of related tasks by using a shared representation that exploits commonalities between them. This definition implies that a multi-task method should be able to deal with problems where different prediction tasks do not necessarily share the same set of training examples or descriptive features and, moreover, each task can have a different data type. Thus, multi-task learning is actually a generalization of MTR.

Artificial neural networks (ANNs) are very well suited for multi-task problems because they can be naturally extended to support multiple outputs and offer flexibility in defining how inputs are shared between tasks. Thus, it is not surprising that most of the earliest multi-task methods were based on ANNs. Caruana (1994), for example, proposed a method where backpropagation is used to train single ANN with multiple outputs (connected to the same hidden layers), and showed that it has better generalization performance compared to multiple single-task ANNs. A different architecture was used by Baxter (1995) where only the first hidden layers are shared and subsequent layers are specific to each task. The question of how much sharing is better when multi-task ANNs are applied for stock return prediction was explored by Ghosn and Bengio (1996) who concluded that a partial sharing of network parameters is preferable compared to full or no sharing. More recently, Collobert and Weston (2008) applied a deep multi-task neural network architecture for natural language processing.

A large number of multi-task learning methods stem from a regularization perspective.[6] Regularization-based multi-task methods minimize a penalized empirical loss of the form $\min_W \mathscr{L}(W) + \Omega(W)$, where W is a parameter matrix that has to be estimated, $\mathscr{L}(W)$ is an empirical loss calculated on the training data and $\Omega(W)$ is a regularization term that takes a different form in each method depending on the underlying task relatedness assumption. Most methods assume that all tasks are related to each other (Evgeniou and Pontil 2004; Ando and Zhang 2005; Argyriou et al. 2006, 2008; Chen et al. 2009, 2010a; Obozinski et al. 2010), while there are methods assuming that tasks are organized in structures such as clusters

[6] A nice categorization of regularization-based multi-task methods can be found in Zhou et al. (2012).

(Jacob et al. 2008; Zhou et al. 2011a), trees (Kim and Xing 2010) and graphs (Chen et al. 2010b). A well-studied category of methods, which are particularly useful when dealing with high-dimensional input spaces, assume that models for different tasks share a common low-rank subspace and impose a trace-norm constraint on the parameter matrix (Argyriou et al. 2006, 2008; Ji and Ye 2009). A similar category of methods constraint all models to share a common set of features (thus performing a joint feature selection), typically by applying L_1/L_q-norm ($q > 1$) regularization (Obozinski et al. 2010). An approach that relaxes the above restrictive constraint allowing models to leverage different extents of feature sharing is proposed in Jalali et al. (2010, 2013).

Finally, we would like to mention that a number of MTR methods are based on the Gaussian Processes framework (e.g., Bonilla et al. 2007; Álvarez and Lawrence 2011). These methods capture correlations between tasks by appropriate choices of covariance functions. A nice review of such methods as well as their relations to regularization-based multi-task approaches can be found in Alvarez et al. (2011).

3.2 Discrepancy in meta-inputs

In the MLC domain, Senge et al. (2013a) studied how the discrepancy issue affects the performance of Classifier Chains and showed that longer chains (i.e. multi-label problems with more labels to be predicted) lead to a higher performance deterioration. In an extension of that work Senge et al. (2013b), a "rectified" version of Classifier Chains (called Nested Stacking) was presented that uses in-sample estimates of the label variables for training as in Stacked Binary Relevance. It was shown that this method performs better than the original Classifier Chains, especially when the label dependencies are strong. Following the opposite direction, Montañés et al. (2011) proposed AID, a method similar to Stacked Binary Relevance, and found that using the actual label values instead of (in-sample) estimates, leads to better results for most multi-label evaluation measures in both AID and Stacked Binary Relevance.

Our work is the first to study this issue in the MTR domain.[7] The issue is studied jointly for SST and ERC, thus allowing general conclusions to be drawn for this type of methods. Furthermore, Montañés et al. (2011), Senge et al. (2013b) compared only the use of actual target values with the use of in-sample estimates while our comparison includes the use of out-of-sample estimates obtained by a cross-validation procedure. Finally, Senge et al. (2013b) evaluate the use of estimates in Classifier Chains whereas we focus on the ensemble version of the corresponding MTR method (ERC) that is expected to offer more resilience to error propagation, as discussed in Sect. 2.5.

4 Experimental setup

This section describes our experimental setup. We first present the participating methods and their parameters and provide details about their implementation in order to facilitate reproducibility of the experiments. Next, we describe the evaluation measure and explain the process that was followed for the statistical comparison of the methods. Finally, we present the datasets that we used and their main statistics.

[7] Actually, an early version of this work Spyromitros-Xioufis et al. (2012) is the first to consider the discrepancy problem in the context of input space expansion methods.

Table 2 Methods used in experiments with abbreviations and citations

Abbr.	Method	Citation
ST	Single target	
SST_{true}	Stacked ST, true values	This paper
SST_{train}	Stacked ST, in-sample estimates	This paper
SST_{cv}	Stacked ST, cv estimates	This paper
ERC_{true}	Ensemble of Regressor Chains, true values	This paper
ERC_{train}	Ensemble of Regressor Chains, in-sample estimates	This paper
ERC_{cv}	Ensemble of Regressor Chains, cv estimates	This paper
MORF	Multi-Objective Random Forest	Kocev et al. (2007)
TNR	Trace Norm Regularization multi-task learning	Argyriou et al. (2008)
DIRTY	A Dirty model for multi-task learning	Jalali et al. (2010, 2013)
RLC	Random Linear target Combinations	Tsoumakas et al. (2014)

4.1 Methods, parameters and implementation

The experimental evaluation includes all variants of the proposed SST and ERC methods, the ST baseline and the following state-of-the-art multiple prediction methods: (a) multi-objective random forest (MORF) (Kocev et al. 2007), (b) trace norm regularization for multi-task learning (TNR) (Argyriou et al. 2008), (c) the DIRTY approach for multi-task learning (Jalali et al. 2010, 2013) and (d) a very recent multi-target method based on random linear combinations of the output space (RLC) (Tsoumakas et al. 2014). For easy reference, Table 2 lists all methods included in the evaluation along with their abbreviations and citations where appropriate.

The proposed methods as well as ST and RLC transform the mutli-target regression task into a series of single-target regression tasks which can be dealt with using any standard regression algorithm. For most of the experiments, we use bagged regression trees as the base regressor. This choice was motivated in Sect. 2.3 and is further discussed in Sect. 5.1 where we present results using a variety of well-known linear and non-linear regression algorithms. The ensemble size of all ERC variants is set to $k = 10$ RC models, each one trained using a different random chain. In datasets with less than 10 distinct chains, we create exactly as many RC models as the number of distinct chains. Furthermore, since the base regressor involves bootstrap sampling, we do not perform sampling in ERC, i.e. each RC model is trained using all training examples. In SST, we exclude the target being predicted by each second stage model from the input space of that model as we found that this choice improves slightly the performance of all variants of this method. $f = 10$ internal cross-validation folds are used in both SST_{cv} and ERC_{cv}.

Concerning the parameter settings of the competitive methods, in MORF we use an ensemble size of 100 trees and the values suggested by Kocev et al. (2007) for the rest of its parameters. In RLC, we generate $r = 100$ new target variables by combining $k = 2$ of the original target variables (after bringing them to the [0, 1] interval). As shown in Tsoumakas et al. (2014), these values lead to near optimal results. In TNR, we minimize the squared loss function using the accelerated gradient method for trace norm minimization (Ji and Ye 2009). The regularization parameter is tuned by selecting among the values $\{10^r : r \in \{-3, \dots, 3\}\}$ with internal 5-fold cross-validation. Before applying TNR, we apply z-score normalization and add a bias column as suggested in Zhou et al. (2011b). Finally, DIRTY is setup as suggested in Jalali et al. (2013): Input variables are scaled to the $[-1, 1]$ range by dividing them with their maximum values. The regularization parameters λ_b and λ_s are tuned via internal 5-fold cross-validation (as in TNR). As suggested in Jalali et al. (2013), we set $\lambda_b = c\sqrt{\frac{m \log d}{n}}$, where $c \in \{10^r : r \in \{-2, \ldots, 2\}\}$ is a constant. Each distinct value of λ_b is paired with five values of $\lambda_s = \frac{\lambda_b}{1+\frac{m-1}{4}i}$, $i \in \{0, 1, 2, 3, 4\}$, thus respecting the $\frac{\lambda_s}{\lambda_b} \in [\frac{1}{m}, 1]$ relationship dictated by the optimality conditions. In total, 25 different combinations of λ_b and λ_s are evaluated.

All the proposed methods and the evaluation framework were implemented in Java and integrated in Mulan[8] (Tsoumakas et al. 2011) by expanding its functionality to multi-target regression. The implementation of all single-target regression algorithms that were used to instantiate problem transformation methods are taken from Weka.[9] With respect to the competing methods, RLC was already integrated in Mulan while for the purposes of this study we also integrated MORF (via a wrapper of the implementation offered in CLUS[10]) as well as TNR and DIRTY (via wrappers of the implementations offered in MALSAR (Zhou et al. 2011b)). Thus, all methods were evaluated under a common framework. In support of open science, we created a github project[11] that contains all our implementations, including code that facilitates easy replication of our experimental results.

4.2 Evaluation

The proposed methods aim at reducing the prediction error on every single target of a MTR problem. To measure the performance of a MTR model on each target variable we use Relative Root Mean Squared Error (RRMSE). The RRMSE of a model **h** that has been induced from a train set D_{train} is estimated based on a test set D_{test} according to the following equation:

$$RRMSE(\mathbf{h}, D_{test}) = \sqrt{\frac{\sum_{(\mathbf{x},\mathbf{y}) \in D_{test}} (\hat{y}_j - y_j)^2}{\sum_{(\mathbf{x},\mathbf{y}) \in D_{test}} (\bar{Y}_j - y_j)^2}} \quad (1)$$

where $\bar{Y}_j$ is the mean value of target variable Y_j over D_{train} and $\hat{y}_j$ is the estimation of the MTR model **h** for Y_j. More intuitively, RRMSE for a target is equal to the Root Mean Squared Error (RMSE) for that target divided by the RMSE of predicting the average value of that target in the training set. RRMSE is estimated using k-fold cross-validation on all datasets, i.e. one RRMSE measurement is obtained on each fold and the final RRMSE is calculated as the average of those measurements. We use $k = 10$ on all datasets, except those with more than 9000 examples where for computational reasons we use either $k = 5$ (rf1 and rf2) or $k = 2$ (scm1d and scm20d).[12]

To test the statistical significance of the observed differences between the methods, we follow the methodology suggested by Demsar (2006). To compare multiple methods on multiple datasets we use the Friedman test, the non-parametric alternative of the repeated-measures ANOVA. The Friedman test operates on the average ranks of the methods and

[8] http://mulan.sourceforge.net.

[9] http://www.cs.waikato.ac.nz/ml/weka.

[10] http://dtai.cs.kuleuven.be/clus/.

[11] https://github.com/lefman/mulan-extended.

[12] The reliability of the estimates obtained using $k = 2$ and $k = 5$ has been validated by checking the stability of the rankings of the methods when repeating the cross-validation experiment with different random seeds.

checks the validity of the hypothesis (null-hypothesis) that all methods are equivalent. Here, we use an improved (less conservative) version of the test that uses the F_f instead of the χ_F^2 statistic (Iman and Davenport 1980). When the null-hypothesis of the Friedman test is rejected ($p < 0.01$), we proceed with the Nemenyi post-hoc test that compares all methods to each other in order to find which methods in particular differ from each other. Instead of reporting the outcomes of all pairwise comparisons, we employ the simple graphical presentation of the test's results introduced by Demsar (2006), i.e. all methods being compared are placed in a horizontal axis according to their average ranks and groups of methods that are not significantly different (at a certain significance level) are connected (see Fig. 4 for an example). To generate such a diagram, a critical difference (CD) should be calculated that corresponds to the minimum difference in average ranks required for two methods to be considered significantly different. CD for a given number of methods and datasets, depends on the desired significance level. Due to the known conservancy of the Nemenyi test (Demsar 2006), we use a 0.05 significance level for computing the CD throughout the paper.

As the above methodology requires a single performance measurement for each method on each dataset, it is not directly applicable to multi-target evaluation where we have multiple performance measurements (one for each target) for each method on each dataset. One option is to take the average RRMSE (aRRMSE) across all target variables within a dataset as a single performance measurement. This choice, however, has the disadvantage that a very small or large error on a single target might dominate the average, thus obscuring performance differences on the target level. Another option is to treat the RRMSE of each method on each target as a different performance measurement. In this case, Friedman test's assumption of independence between performance measurements might be violated. In the absence of a better solution, we perform a two-dimensional analysis (as done e.g. by Aho et al. 2012) where statistical tests are conducted using both aRRMSE (*per dataset analysis*) but also considering RRMSE per target as an independent performance measurement (*per target analysis*).

4.3 Datasets

Despite the numerous interesting applications of MTR, there are only few publicly available datasets of this kind—perhaps because most applications are industrial—and most experimental evaluations of MTR methods are based on a limited amount of datasets. For this study, much effort was made for the composition of a large and diverse collection of benchmark MTR datasets. In addition to 5 datasets that have been used in previous studies and are publicly available (edm, sf1, sf2, jura, wq), we also used 5 publicly available datasets (enb, slump, andro, osales, scpf) that have not been used for MTR benchmarking in the past. We also collected raw MTR data from a variety of interesting application domains and composed 8 new benchmark datasets (atp1d, atp7d, oes97, oes10, rf1, rf2, scm1d, scm20d). In total we collected 18 datasets and make them publicly available for future studies.[13] To the best of our knowledge, this is the largest collection of benchmark MTR datasets to date.

Table 3 reports the name (1st column), source (2nd column), number of examples (3rd column), number of input variables (4th column) and number of target variables (5th column) of each dataset. Detailed descriptions of all datasets are provided in "Appendix".

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

Table 3 Name, source, number of examples, number of input variables (d) and number of target variables (m) of the datasets used in the evaluation

Dataset	Source	Examples	d	m
edm	Karalic and Bratko (1997)	154	16	2
sf1	Lichman (2013)	323	10	3
sf2	Lichman (2013)	1066	10	3
jura	Goovaerts (1997)	359	15	3
wq	Dzeroski et al. (2000)	1060	16	14
*enb	Tsanas and Xifara (2012)	768	8	2
*slump	Yeh (2007)	103	7	3
*andro	Hatzikos et al. (2008)	49	30	6
*osales	Kaggle (2012)	639	413	12
*scpf	Kaggle (2013)	1137	23	3
*atp1d	This paper	337	411	6
*atp7d	This paper	296	411	6
*oes97	This paper	334	263	16
*oes10	This paper	403	298	16
*rf1	This paper	9125	64	8
*rf2	This paper	9125	576	8
*scm1d	This paper	9803	280	16
*scm20d	This paper	8966	61	16

The datasets marked with an asterisk are first used for MTR benchmarking in this paper to the best of our knowledge

5 Experimental analysis

In this section we present an extensive experimental analysis of the performance of the proposed methods. Sect. 5.1 is devoted to an exploration of the performance of ST using various well-known regression algorithms. The purpose of this investigation is to help us select an algorithm that works well on the studied datasets and use it as base regressor in all problem transformation methods (ST, SST, ERC and RLC) in subsequent experiments. At the same time, a challenging baseline performance level will be set for all multi-target methods. In Sect. 5.2 we evaluate SST$_{train}$ and ERC$_{true}$, the direct adaptations of the corresponding MLC methods, in order to see whether these variants obtain a competitive performance compared to ST and state-of-the-art multi-target methods. Next, in Sect. 5.3 all three meta-input generation variants ($true$, $train$, cv) of SST and ERC are evaluated and compared to ST, shedding light into the impact of the discrepancy problem on each method. After the best performing variants of each method have been identified, Sect. 5.4 compares them with the state-of-the-art. The running times of all methods are reported and compared in Sect. 5.5, and finally, this section ends with a discussion of the main outcomes of the experimental results (Sect. 5.6).

5.1 Base regressor exploration

In this subsection we explore the performance of ST on the studied domains using a variety of regression algorithms. The goal of this exploration is to help us identify a regression algorithm that performs well across many domains, thus setting a challenging baseline performance level for the multi-target methods that we study next. The algorithm that will emerge as the best performer will be used to instantiate all problem transformation methods (ST, SST,

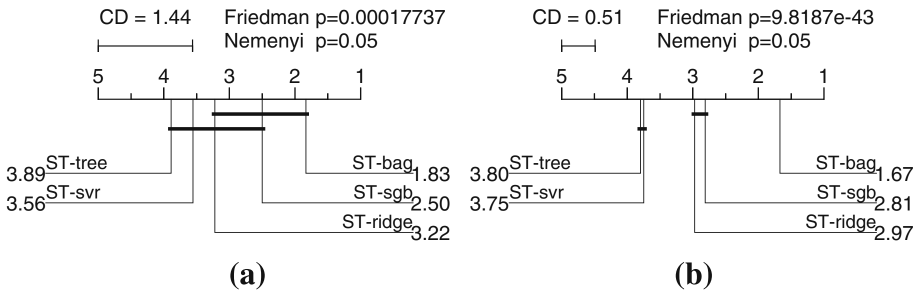


Fig. 3 Comparison of different ST instantiations using the Nemenyi test. Groups of methods that are not significantly different (at $p = 0.05$) are connected. **a** Per dataset analysis. **b** Per target analysis

ERC and RLC) in the rest of the experiments, facilitating a fair comparison between these methods.

We selected five well-known linear and non-linear regression algorithms to couple ST with, in particular we use: ridge regression (Hoerl and Kennard 1970) (RIDGE), regression tree Breiman et al. (1984) (TREE), L2-regularized support vector regression regression (Drucker et al. 1996) (SVR), bagged (Breiman 1996) regression trees (BAG) and stochastic gradient boosting (Friedman 2002) (SGB). In RIDGE and SVR, the regularization parameter was tuned (separately for each target) by applying internal 5-fold cross-validation and choosing the value that leads to the lowest root mean squared error among $\{10^r : r \in \{-4, \dots, 2\}\}$. In BAG we combine the predictions of 100 TREEs while in SGB we boost trees with four terminal nodes using a small shrinkage rate (0.1) and a large number of iterations (100), as suggested by Friedman et al. (2001).

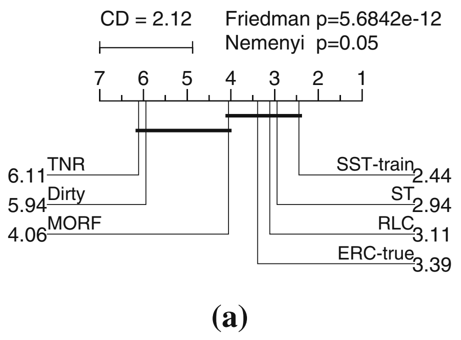
The detailed results obtained by each instantiation on each dataset and target are given in Appendix "Base regressor exploration results". We observe that no algorithm is better in all domains (as dictated by the no free lunch theorems for supervised learning (Wolpert 1996, 2002)). However, ST-BAG stands out obtaining the lowest aRRMSE in nine datasets. ST-SGB follows with five wins while ST-RIDGE and ST-SVR each obtain the lowest error in two datasets. Figure 3 shows the average ranks of the different instantiations along with the results of the Friedman and the Nemenyi tests for the analysis per dataset (left) and per target (right). In both analyses, the lowest average rank is obtained by ST-BAG, followed by ST-SGB and ST-RIDGE. In the per dataset analysis, the Nemenyi test finds that ST-BAG is significantly better than ST-TREE and ST-SVR while in the per target analysis, ST-BAG is found significantly better than all the other instantiations. Therefore, we use BAG as the base regressor for all problem transformation methods in the rest of the experiments.

5.2 Evaluation of direct adaptations

In this subsection we focus on SST$_{train}$ and ERC$_{true}$, the versions of SST and ERC that use the same type of values for the meta-inputs as their MLC counterparts, and compare their performance to that of ST, MORF, RLC, TNR and DIRTY to see where these methods stand with respect to the state-of-the-art.

Figure 4 shows the average ranks of the methods along with the results of the Friedman and the Nemenyi tests when the analysis is performed per dataset (left) and per target (right).[14] Several interesting remarks can be made based on these results. First, we see that *both*

[14] The detailed results per dataset and target can be found in Appendix "Multi-target regression results".



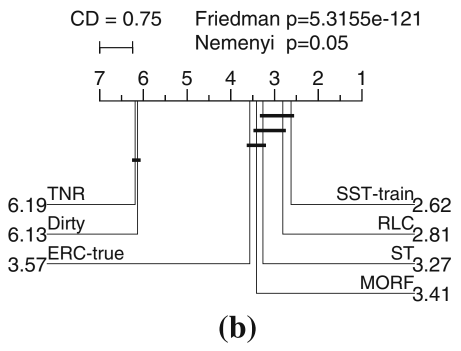


Fig. 4 Comparison of direct adaptations (with BAG as base regressor) using the Nemenyi test. Groups of methods that are not significantly different (at $p = 0.05$) are connected. **a** Per dataset analysis. **b** Per target analysis

SST$_{train}$ and ERC$_{true}$ are competitive with state-of-the-art methods. SST$_{train}$ obtains the lowest average rank in both the per dataset and the per target analysis. In the per dataset analysis, it is found significantly better than TNR and DIRTY and similar with MORF and RLC and in the per target analysis it is found better than TNR, DIRTY and MORF and similar with RLC. ERC$_{true}$ performs worse than SST$_{train}$ but is still ranked above TNR, DIRTY and MORF in the per dataset analysis and above TNR and DIRTY in the per target analysis. In the per dataset analysis, ERC$_{true}$ is found significantly better than TNR and DIRTY and similar with MORF and RLC, while in the per target analysis it is found better than TNR and DIRTY, similar with MORF and only RLC outperforms it significantly.

Interestingly, however, we see that according to both the per dataset and the per target analysis, SST$_{train}$ and ERC$_{true}$ are not significantly better than ST. This is an indication that the use of targets as meta-input as implemented by these variants of SST and ERC does not bring significant improvements. Actually, as can be seen from the detailed results, both SST$_{train}$ and ERC$_{true}$ perform worse than ST in several cases. This issue is studied in more detail in the following subsection.

Perhaps even more interestingly, *none of the state-of-the-art multi-target methods participating in this comparison manages to significantly improve the performance of ST*. In fact, ST is ranked second after SST$_{train}$ in the per dataset analysis and third after SST$_{train}$ and RLC in the per target analysis, and is found significantly better than TNR and DIRTY in both types of analyses. This exceptionally good performance of ST might seem a bit surprising given the results of previous studies (e.g. Kocev et al. 2007; Tsoumakas et al. 2014) but is in accordance with empirical and theoretical results for Binary Relevance (as discussed in Sect. 2) and is attributed to the use of a very strong base regressor.

To validate this, we instantiate all problem transformation methods with RIDGE, a base regressor that was found to perform worse than BAG in Sect. 5.1, and repeat the comparison. As shown in Fig. 5, the situation is quite different compared to when BAG was used as base regressor. We observe that ST is now ranked below MORF, RLC and ERC$_{true}$ in both the per dataset and the per target analysis and is found significantly worse than MORF according to the per target analysis. Clearly, *as the strength of the base regressor increases, i.e. when the information provided by the features is well exploited, improving the performance of ST becomes more difficult*. However, it is this challenging setting where performance improvements matter the most and it is thus interesting to see whether the proposed extensions of SST and ERC manage to obtain more consistent improvements over ST (compared to SST$_{train}$ and ERC$_{true}$) under this setting.

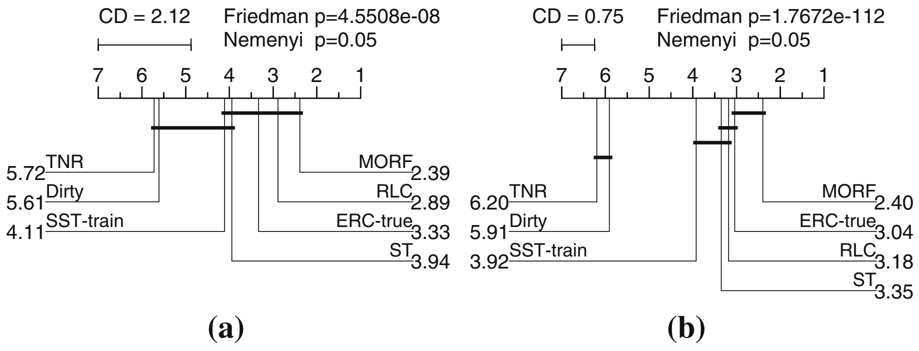


Fig. 5 Comparison of direct adaptations (with RIDGE as base regressor) using the Nemenyi test. Groups of methods that are not significantly different (at $p = 0.05$) are connected. **a** Per dataset analysis. **b** Per target analysis

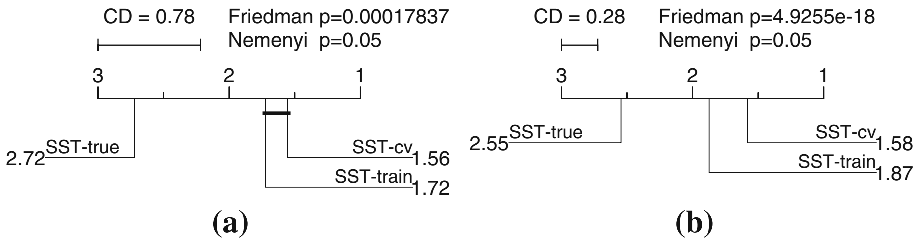


Fig. 6 Comparison of SST variants using the Nemenyi test. Groups of methods that are not significantly different (at $p = 0.05$) are connected. **a** SST, per dataset analysis. **b** SST, per target analysis

5.3 Evaluation of meta-input generation variants

In this subsection we evaluate the performance of SST and ERC when different types of values are used for the meta-inputs at training time. In particular, each method is evaluated using the actual target values ($true$ variants), in-sample estimates ($train$ variants) and out-of-sample estimates (cv variants) generated using the proposed internal cross-validation strategy. We want to see whether the cv variants (that according to the discussion of Sect. 2.4 are expected to be less affected by the discrepancy problem) can indeed perform better than the $train$ and $true$ variants and whether they manage to obtain more consistent improvements over ST. We also want to see how the SST variants compare to the ERC variants.

Figures 6 and 7 show the average ranks and the results of the Friedman and Nemenyi tests for the three variants of SST and ERC, respectively, according to the per dataset (left) and the per target (right) analysis. First, we see that in both SST and ERC and in both types of analyses, the variants that use the actual values of the targets ($true$) obtain the worst average ranks and are found significantly worse than both variants that use estimates ($train$ and cv). Since the variants of each method differ only with respect to the type of values that they use for the meta-inputs, it is clear that *the discrepancy problem has a significant impact on the performance of both SST and ERC and that the use of estimates can ameliorate this problem*.

With respect to the kind of estimates that should be used (in-sample or out-of-sample) the situation is slightly different for each method. In the case of SST, the cv variant obtains the best average rank in both the per dataset and the per target analysis and its difference with the $train$ variant is found significant in the per target analysis. In the case of ERC, while

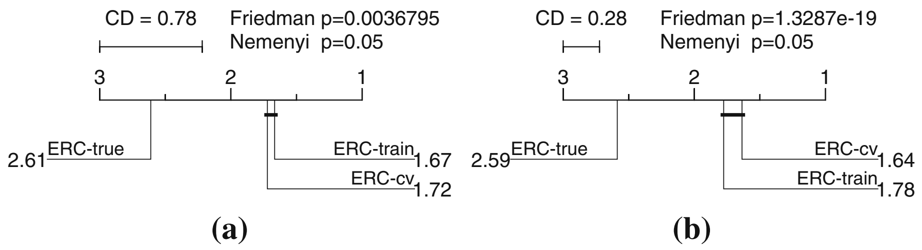


Fig. 7 Comparison of ERC variants using the Nemenyi test. Groups of methods that are not significantly different (at $p = 0.05$) are connected. **a** ERC, per dataset analysis. **b** ERC, per target analysis

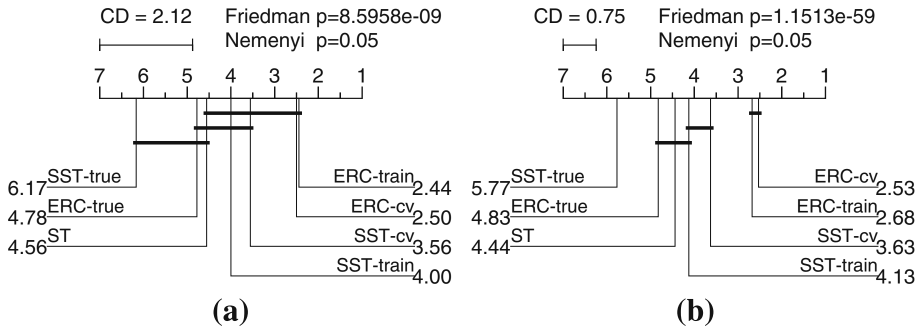


Fig. 8 Comparison of SST$_{true/train/cv}$, ERC$_{true/train/cv}$ and ST using the Nemenyi test. Groups of methods that are not significantly different (at $p = 0.05$) are connected. **a** Per dataset analysis. **b** Per target analysis

the cv variant is ranked higher than the $train$ variant in the per target analysis, the $train$ variant is ranked slightly higher in the per dataset analysis and in both cases the differences are not found significant. This suggests that *using out-of-sample estimates is important for SST while ERC seems to be less affected by the discrepancy problem and, as a result, the use of in-sample estimates can be considered as a viable alternative*.

A question that has not been answered yet, is how the new variants of SST and ERC compare to ST and to each other. Figure 8 shows the results of the Friedman and the Nemenyi tests when all variants of SST and ERC are compared together with ST. We see that in both the per dataset (left) and the per target (right) analysis, the four variants that use estimates for the meta-inputs obtain lower average ranks than ST while the $true$ variants obtain worse average ranks. The differences with ST are not found significant according to the per dataset analysis but according to the per target analysis ERC$_{train}$ and ERC$_{cv}$ are found significantly better. Comparing the SST variants with the ERC variants, we see that each ERC variant is always ranked above the corresponding SST variant. This suggests that *ERC's strategy for leveraging information from target variables is beneficial*. Moreover, we see that that ERC$_{train}$ and ERC$_{cv}$ *are found significantly better than the rest of the methods* according to the per target analysis.

5.3.1 Cautiousness analysis

So far, our analysis has focused on the average performance of the proposed methods (as quantified by their average ranks over datasets and targets) and we found that ERC$_{train}$ and ERC$_{cv}$ outperform the independent regressions baseline significantly. However, it is also important to see the consistency of these improvements across different datasets and targets. In particular, we would like to study the degree of cautiousness that each method exhibits, i.e. how frequently and to what extent are the predictions produced by each method less accurate than the predictions of ST.

To facilitate a comparison of the methods in this regard, the following measures are defined:

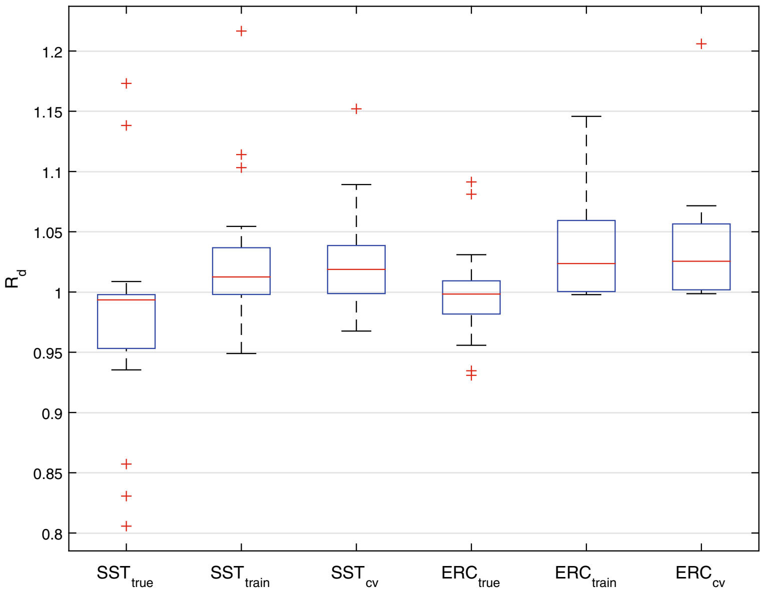
$$R_d(M) = \frac{aRRMSE(ST)}{aRRMSE(M)},$$
$$R_t(M) = \frac{RRMSE(ST)}{RRMSE(M)}.$$

For each method M and dataset d, R_d quantifies the amount of improvement or degradation induced by M compared to ST in terms of $aRRMSE$. Similarly, for each method M and target t, R_t quantifies the amount of improvement or degradation compared to ST in terms of $RRMSE$. Values of $R_d(M)$ and $R_t(M) < 1$ indicate that the method produces worse estimates than ST ($R_d(ST) = R_t(ST) = 1$). The upper part of Fig. 9 displays box plots of the values of R_d over the 18 datasets included in the experimental study, i.e. each box plot summarizes the distribution of 18 values, while the lower part displays box plots of the values of R_t over 143 targets, i.e. each box plot summarizes the distribution of 143 values.

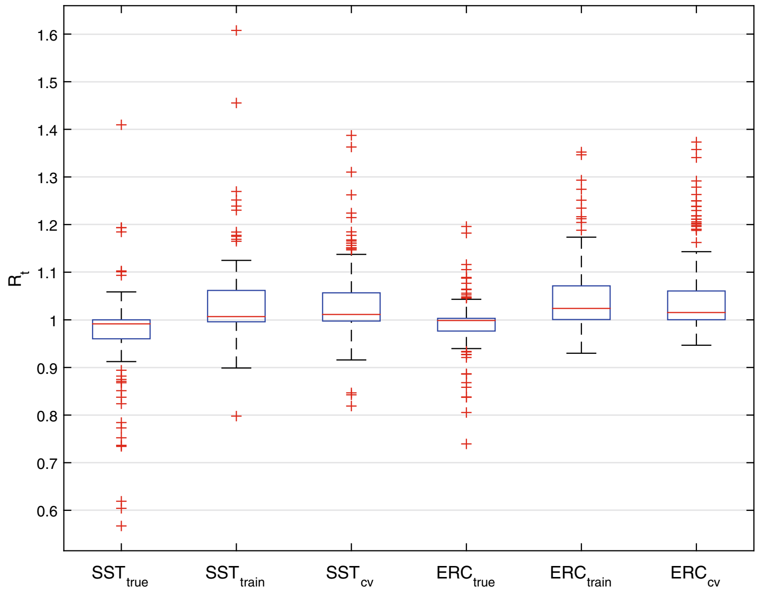
We see that that in both the per dataset and the per target analysis, the $true$ variants are the ones exhibiting the more dispersed distributions with several cases of significant degradation of ST's performance. The $train$ and cv variants are clearly more cautious with much fewer cases of degradation and even fewer cases of significant degradation. Looking at the distributions of R_t, we could say that *the cv variants appear a bit more cautious than the $train$ variants especially in the case of SST*. We also see that the ERC variants are always more cautious than the corresponding SST variants. Clearly, ERC$_{train}$ and ERC$_{cv}$ are the two most cautious methods since they obtain very similar or better performance than ST on all datasets and on about 75 % of the targets. Even on targets where the two methods obtain a lower performance than ST, the reduction is less than about 5 %. This characteristic along with the fact that they obtain the largest average improvements over ST, make ERC$_{train}$ and ERC$_{cv}$ highly appealing.

5.4 Comparison with the state-of-the-art

In this section we compare the three best performing variants of the proposed methods, i.e. ERC$_{cv}$, ERC$_{train}$ and SST$_{cv}$, with MORF, RLC, TNR and Dirty to see how they compare to the state-of-the-art. Figure 10 shows the results of the Friedman and Nemenyi tests for the analysis per dataset (left) and per target (right). The per dataset analysis shows that all our methods perform significantly better than TNR and Dirty while ERC$_{cv}$ and ERC$_{train}$ also perform significantly better than MORF. Moreover, all our methods obtain a lower average rank than RLC but according to this analysis the differences are not significant. According to the per target analysis, all our methods are found significantly better than TNR, Dirty and MORF, and additionally, ERC$_{cv}$ and ERC$_{train}$ are found significantly better than RLC. In Fig. 11 we compare the performance of the methods from a cautiousness perspective, as we did in Sect. 5.3. TNR, Dirty and MORF are far less cautious than SST$_{cv}$, ERC$_{train}$ and ERC$_{cv}$ with many instances of extreme degradation of ST's performance. RLC is more cautious but not as much as SST$_{cv}$, ERC$_{train}$ and ERC$_{cv}$, especially according to the per target analysis.



(a)



(b)

Fig. 9 Cautiousness analysis of SST and ERC variants. On *each box*, the *whiskers* extend to the most extreme data points still within 1.5 IQR of the lower quartile and outliers are plotted individually. **a** Distribution of R_d values for each method over 18 datasets. **b** Distribution of R_t values for each method over 143 targets

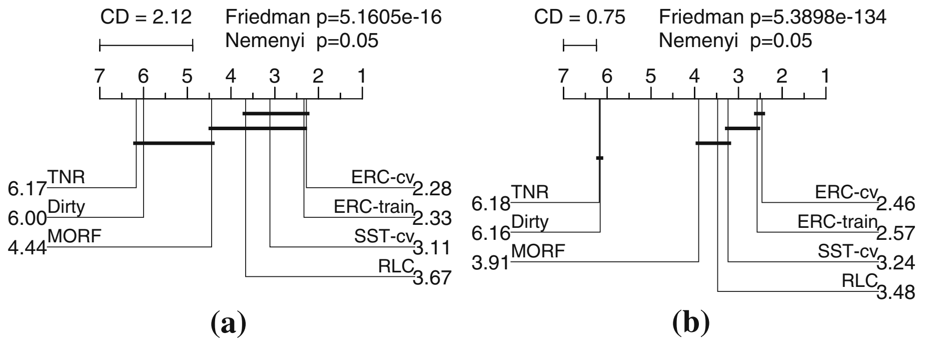


Fig. 10 Comparison of the best SST and ERC variants with the state-of-the-art using the Nemenyi test. Groups of methods that are not significantly different (at $p = 0.05$) are connected. **a** Per dataset analysis. **b** Per target analysis

5.5 Running times

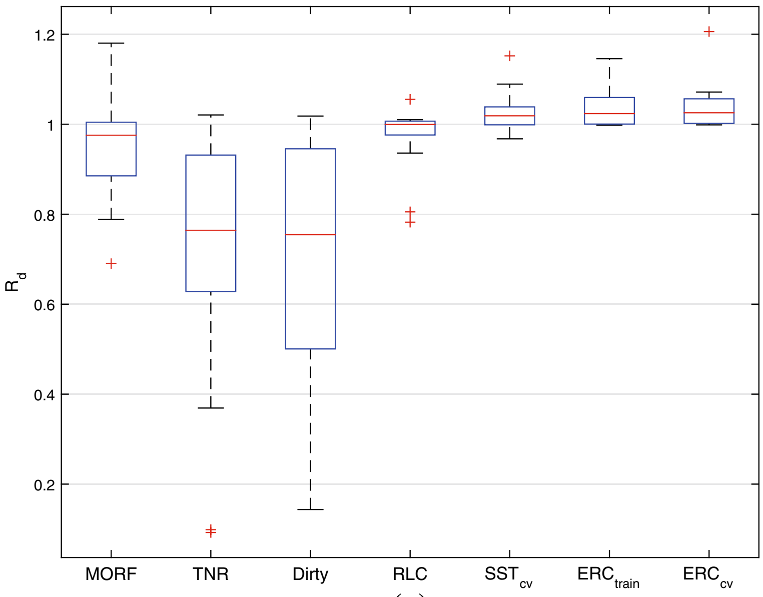
In this subsection we compare the running times of the studied methods. Experiments were run on a 64-bit CentOS Linux machine with 80 Intel Xeon E7-4860 processors running at 2.27 GHz and 1 TB of main memory. The detailed results per method and dataset are shown in Table 4. For ST, RLC, SST and ERC we report times with BAG as base regressor. The number shown in parenthesis next to the name of each dataset corresponds to the maximum number of processor threads that were available during the experiment. ST, SST, ERC and RLC made use of multiple threads through Weka's multi-threaded implementation of Bagging. Thus, running times are directly comparable for these methods. Multi-threading was also partly used in TNR for the computation of the gradients. DIRTY and MORF, on the other hand, always used a single processor thread.

Looking at the aggregated running times, we see that MORF is by far the most efficient method, followed by ST, SST$_{true}$ and SST$_{train}$ which have similar running times. On the other hand, DIRTY is the least efficient method, followed by ERC$_{cv}$. The running times of the rest of the methods lie in between. With respect to the SST and ERC variants, we see that their running times agree with the complexity analysis of Sect. 2.6. The total running time of SST$_{true}$ is roughly twice the total running time of ST and similar to the total running time of SST$_{train}$. SST$_{cv}$ is the least efficient among SST variants with a total running time that is about 5 times larger than that of SST$_{true}$ and SST$_{train}$. With respect to the ERC variants, we see that ERC$_{true}$ and ERC$_{train}$ have similar total running times (which are also roughly similar to the total running time of SST$_{cv}$) while ERC$_{cv}$ is about 7.5 times slower.

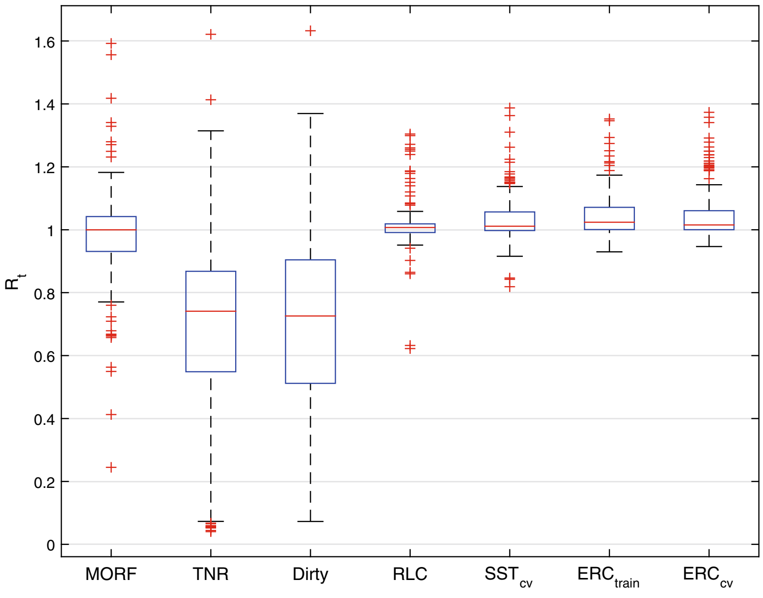
Overall, we see that the improvements achieved by ERC$_{cv}$ and ERC$_{train}$ over ST come with an increased computational cost. However, this cost is manageable especially in the case of ERC$_{train}$. Furthermore, when better efficiency is needed, besides the use of parallelization one might consider reducing the ensemble size (k) or using a smaller number of folds (f) when applying internal cross-validation (in ERC$_{cv}$).

5.6 Discussion

Several interesting conclusions can be drawn from our experimental results. The experiments of Sects. 5.2 and 5.3 showed that while the directly adapted versions of SST and ERC have comparable or better performance than state-of-the-art methods, a careful handling of the



(a)



(b)

Fig. 11 Comparing the cautiousness of the best SST and ERC variants to that of state-of-the-art methods. On *each box*, the whiskers extend to the most extreme data points still within 1.5 IQR of the lower quartile and outliers are plotted individually. **a** Distribution of R_d values for each method over 18 datasets. **b** Distribution of R_t values for each method over 143 targets

Table 4 Running times (in s) using BAG as base regressor

Dataset	ST	SST_{true}	SST_{train}	SST_{cv}	ERC_{true}	ERC_{train}	ERC_{cv}	MORF	RLC	TNR	DIRTY
edm (2)	4.3	3.3	3.2	14.5	**2.8**	2.8	13.7	6.3	78.3	16.6	281.4
sf1 (3)	4.1	**4.1**	4.3	16.7	11.2	11.8	60.8	8.1	79.2	11.8	60.5
sf2 (3)	**7.5**	11.5	11.4	47.6	33.2	36.3	262.0	15.4	238.7	17.3	56.9
jura (3)	**10.9**	15.3	14.9	72.6	46.4	46.9	264.2	14.0	199.1	17.0	49.2
wq (6)	**43.3**	104.0	141.7	540.0	561.3	607.0	4648.4	106.3	295.0	143.7	531.2
enb (2)	**10.4**	15.4	15.9	76.0	16.7	16.4	74.9	15.7	313.9	35.8	223.0
slump (3)	4.7	4.2	**4.0**	16.5	11.4	11.1	67.8	4.2	48.5	10.7	55.6
andro (2)	8.2	10.6	9.0	47.9	49.5	47.2	360.3	**3.1**	72.5	445.3	1670.8
onsales (8)	317.6	612.2	560.8	2803.0	2835.0	2801.6	21568.6	**43.2**	1628.0	2769.1	80616.2
scpf (3)	22.9	36.2	36.6	196.5	106.2	106.4	651.2	**13.4**	491.3	84.5	215.4
atp1d (4)	155.6	356.1	345.7	1614.7	1496.0	1555.6	12169.2	**23.3**	2408.0	2212.8	57276.5
atp7d (4)	145.8	318.6	313.9	1517.2	1336.9	1385.0	11158.2	**18.4**	2179.2	1536.0	53974.0
oes97 (6)	199.5	454.8	422.2	1998.4	1977.3	1965.5	16609.1	**32.7**	1057.1	4581.3	124231.5
oes10 (6)	286.4	613.1	535.7	2725.9	2691.0	2575.1	21446.6	**39.4**	1182.6	6093.8	157399.6
rf1 (8)	379.6	868.2	890.8	4180.7	3885.7	4018.8	30050.1	**351.1**	4184.7	2952.1	34927.2
rf2 (10)	3505.5	6539.0	6065.6	31562.8	28347.1	28104.0	234088.5	**325.2**	35429.1	18130.9	197482.1
scm1d (10)	1398.5	2499.5	2398.3	11111.9	10465.6	11253.2	113500.0	**199.7**	6018.6	5647.1	105215.8
scm20d (8)	302.6	613.9	621.8	2615.8	2733.1	2646.9	18549.2	**140.8**	1627.4	1120.2	4779.6
Total	6807.6	13079.8	12395.8	61158.7	56606.3	57191.6	485542.9	**1360.2**	57531.3	45826.0	819046.5

The number in parenthesis next to the name of each dataset corresponds to the maximum number of processor threads that could be utilized during the experiment discrepancy problem is crucial for obtaining consistent improvements over the independent regressions baseline and the state-of-the-art. In particular, as the experiments of Sect. 5.3 revealed, the use of estimates for the meta-inputs during training should clearly be preferred over using the actual target values. With regard to using in-sample versus out-of-sample estimates, the results indicate that while out-of-sample estimates are preferable in SST, ERC performs almost equally well using either type of estimates for the meta-inputs. As discussed in Sect. 2.5, ERC's models are built on input spaces which are expanded with fewer meta-inputs compared to SST's models and, as a result, a smaller amount of error accumulation is risked at prediction time.

Another interesting conclusion is that when a strong base regressor is employed, the task of improving the performance of ST becomes very difficult. As a result, multi-target methods which are considered state-of-the-art fail to improve ST's performance and are even performing significantly worse. This was particularly the case for the two multi-task methods, TNR and DIRTY, which were consistently found to be the worst performers. One explanation for their bad performance is the fact that both methods are based on a linear formulation of the problem that, as revealed by the base regressor exploration experiments, is not the most suitable hypothesis representation for the studied datasets (RIDGE and SVR performed worse than SGB and BAG that are based on a non-linear hypothesis representation). Moreover, multi-task methods are expected to work better than single-task methods in cases where there is a lack of training data for some of the tasks (Alvarez et al. 2011). This is not the case for most of the datasets that we used in this study as well as many recent multi-target prediction problems. In fact, the two datasets where TNR and DIRTY perform better than ST (sf1 and slump) are among those with the fewest training examples.

With respect to MORF, although it was found significantly more competitive than TNR and DIRTY, it also performed worse than ST on average. Nevertheless, we should point out that MORF achieved the best accuracy on three datasets (edm, wq, andro) and is the most computationally efficient of the compared methods. Similarly to TNR and DIRTY, MORF has the disadvantage of having a fixed hypothesis representation (trees), as opposed to the proposed methods that have the ability of adapting better to a specific domain by being instantiated with a more suitable base regressor. This advantage of the proposed methods is shared with RLC which, however, was not found as accurate.

Overall, our experimental results demonstrate that of the methods proposed in this paper, ERC_{train} and ERC_{cv} and, to a lesser extent, SST_{train} and SST_{cv} provide increased accuracy over doing a separate regression per target. In addition, ERC_{train} and ERC_{cv} are significantly more accurate than TNR, DIRTY, MORF and RLC (in the per target analysis). If caution is a further concern, then again ERC_{train} and ERC_{cv} compare favorably to the rest of the methods. With respect to the $true$ variants of SST and ERC, we should stress out that despite having a worse average performance, they are worthy of being considered by a practitioner as they obtain the highest performance in datasets (e.g., sf1 and scfp) where the discrepancy problem is not predominant.

6 Conclusion

Motivated by the similarity between the tasks of multi-label classification and multi-target regression, this paper introduced SST and ERC, two new multi-target regression techniques derived through a simple adaptation of two well-known multi-label classification methods. Both methods are based on the idea of treating other prediction targets as additional input variables, and represent a conceptually simple way of exploiting target dependencies in order to improve prediction accuracy.

A comprehensive experimental analysis that includes a multitude of real-world datasets and four existing state-of-the-art methods, reveals that, despite being competitive with the state-of-the-art, the directly adapted versions of SST and ERC do not manage to obtain significant improvements or even degrade the performance of the independent regressions baseline. This degradation is attributed to an underestimation (in the original formulations of the methods) of the impact of the discrepancy of the values used for the additional input variables between training and prediction. Confirming our hypothesis, extended versions of the methods that attempt to mitigate the discrepancy using out-of-sample estimates of the targets during training, manage to obtain consistent and significant improvements over the baseline approach and are found significantly better than four state-of-the-art methods. The fact that these impressive results were obtained by applying relatively simple adaptations of existing multi-label classification methods, highlights the importance of exploiting relationships between similar machine learning tasks.

Concluding, let us point to some directions for future work. Although a mitigation of the discrepancy problem leads to significant performance improvements, a different amount of mitigation is ideal for each target. As a result, the use of in-sample estimates (or even the actual target values) gives better results for some targets. Thus, a promising direction for future work would be a deeper theoretical analysis of the different variants and the identification of problem characteristics that favor the use of one variant over the other. Finally, we should point out that SST and ERC can be viewed as strategies for leveraging variables that are available in the training phase but not in the prediction phase. This type of scenario is very common, for instance, in time series prediction. We believe that adapting SST and ERC for this type of problems is another valuable opportunity for future work.

Appendix 1: Datasets

Existing datasets

EDM

The Electrical Discharge Machining dataset (Karalic and Bratko 1997) represents a two-target regression problem. The task is to shorten the machining time by reproducing the behaviour of a human operator that controls the values of two variables. Each of the target variables takes 3 distinct numeric values ($\{-1, 0, 1\}$) and there are 16 continuous input variables.

SF

The Solar Flare dataset (Lichman 2013) has 3 target variables that correspond to the number of times 3 types of solar flare (common, moderate, severe) are observed within 24 h. There are two versions of this dataset. SF1 contains data from year 1969 and SF2 from year 1978.

JURA

The Jura (Goovaerts 1997) dataset consists of measurements of concentrations of seven heavy metals (cadmium, cobalt, chromium, copper, nickel, lead, and zinc), recorded at 359 locations in the topsoil of a region of the Swiss Jura. The type of land use (Forest, Pasture, Meadow, Tillage) and rock type (Argovian, Kimmeridgian, Sequanian, Portlandian, Quaternary) were also recorded for each location. In a typical scenario (Goovaerts 1997; Álvarez and Lawrence 2011), we are interested in the prediction of the concentration of metals that are more expensive to measure (primary variables) using measurements of metals that are cheaper to sample (secondary variables). In this study, cadmium, copper and lead are treated as target variables while the remaining metals along with land use type, rock type and the coordinates of each location are used as predictive features.

WQ

The Water Quality dataset (Dzeroski et al. 2000) has 14 target attributes that refer to the relative representation of plant and animal species in Slovenian rivers and 16 input attributes that refer to physical and chemical water quality parameters.

New datasets

ENB

The Energy Building dataset (Tsanas and Xifara 2012) concerns the prediction of the heating load and cooling load requirements of buildings (i.e. energy efficiency) as a function of eight building parameters such as glazing area, roof area, and overall height, amongst others.

SLUMP

The Concrete Slump dataset (Yeh 2007) concerns the prediction of three properties of concrete (slump, flow and compressive strength) as a function of the content of seven concrete ingredients: cement, fly ash, blast furnace slag, water, superplasticizer, coarse aggregate, and fine aggregate.

ANDRO

The Andromeda dataset (Hatzikos et al. 2008) concerns the prediction of future values for six water quality variables (temperature, pH, conductivity, salinity, oxygen, turbidity) in Thermaikos Gulf of Thessaloniki, Greece. Measurements of the target variables are taken from under-water sensors with a sampling interval of 9 seconds and then averaged to get a single measurement for each variable over each day. The specific dataset that we use here corresponds to using a window of 5 days (i.e. features attributes correspond to the values of the six water quality variables up to 5 days in the past) and a lead of 5 days (i.e. we predict the values of each variable 6 days ahead).

OSALES

This is a pre-processed version of the dataset used in Kaggle's "Online Product Sales" competition (Kaggle 2012) that concerns the prediction of the online sales of consumer products. Each row in the dataset corresponds to a different product that is described by various product features as well as features of an advertising campaign. There are 12 target variables corresponding to the monthly sales for the first 12 months after the product launches. For the purposes of this study we removed examples with missing values in any target variable (112 out of 751) and attributes with one distinct value (145 out of 558).

SCPF

This is a pre-processed version of the dataset used in Kaggle's "See Click Predict Fix" competition (Kaggle 2013). It concerns the prediction of three target variables that represent the number of views, clicks and comments that a specific 311 issue will receive. The issues have been collected from 4 cities (Oakland, Richmond, New Haven, Chicago) in the US and span a period of 12 months (01/2012–12/2012). The version of the dataset that we use here is a random 1 % sample of the data. In terms of features we use the number of days that an issues stayed online, the source from where the issue was created (e.g. android, iphone, remote api, etc.), the type of the issue (e.g. graffiti, pothole, trash, etc.), the geographical co-ordinates of the issue, the city it was published from and the distance from the city center. All multi-valued nominal variables were first transformed to binary and then rare binary variables (being true for less than 1 % of the cases) were removed.

OES

The Occupational Employment Survey datasets were obtained from years 1997 (OES97) and 2010 (OES10) of the annual Occupational Employment Survey compiled by the US Bureau of Labor Statistics. Each row provides the estimated number of full-time equivalent employees across many employment types for a specific metropolitan area. There are 334 and 403 cities in the 1997 and 2010 datasets, respectively. The input variables in these datasets are a randomly sequenced subset of employment types (e.g. doctor, dentist, car repair technician, etc.) observed in at least 50 % of the cities (some categories had no values for particular cities). The targets for both years are randomly selected from the entire set of categories above the 50 % threshold. Missing values in both the input and the target variables were replaced by sample means for these results. To our knowledge, this is the first use of the OES dataset for benchmarking of multi-target prediction algorithms.

ATP

The Airline Ticket Price dataset concerns the prediction of airline ticket prices. The rows are a sequence of time-ordered observations over several days. Each sample in this dataset represents a set of observations from a specific observation date and departure date pair. The input variables for each sample are values that may be useful for prediction of the airline ticket prices for a specific departure date. The target variables in these datasets are the next day (ATP1D) price or minimum price observed over the next 7 days (ATP7D) for 6 target flight preferences: (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. The input variables include the following types: the number of days between the observation date and the departure date (1 feature), the boolean variables for day-of-the-week of the observation date (7 features), the complete enumeration of the following 4 values: (1) the minimum price, mean price, and number of quotes from (2) all airlines and from each airline quoting more than 50 % of the observation days (3) for non-stop, one-stop, and two-stop flights, (4) for the current day, previous day, and two days previous. The result is a feature set of 411 variables. For specific details on how these datasets are constructed please consult Groves and Gini (2015). The nature of these datasets is heterogeneous with a mixture of several types of variables including boolean variables, prices, and counts.

RF

The river flow datasets concern the prediction of river network flows for 48 h in the future at specific locations. The dataset contains data from hourly flow observations for 8 sites in the Mississippi River network in the United States and were obtained from the US National Weather Service. Each row includes the most recent observation for each of the 8 sites as well as time-lagged observations from 6, 12, 18, 24, 36, 48 and 60 h in the past. In RF1, each site contributes 8 attribute variables to facilitate prediction. There are a total of 64 variables plus 8 target variables.The RF2 dataset extends the RF1 data by adding precipitation forecast information for each of the 8 sites (expected rainfall reported as discrete values: 0.0, 0.01, 0.25, 1.0 inches). For each observation and gauge site, the precipitation forecast for 6 h windows up to 48 h in the future is added (6, 12, 18, 24, 30, 36, 42, and 48 h). The two datasets both contain over 1 year of hourly observations (>9000 h) collected from September 2011 to September 2012. The domain is a natural candidate for multi-target regression because there are clear physical relationships between readings in the contiguous river network.

SCM

The Supply Chain Management datasets are derived from the Trading Agent Competition in Supply Chain Management (TAC SCM) tournament from 2010. The precise methods for data preprocessing and normalization are described in detail by Groves and Gini (2011). Some benchmark values for prediction accuracy in this domain are available from the TAC SCM Prediction Challenge (Pardoe and Stone 2008), these datasets correspond only to the "Product Future" prediction type. Each row corresponds to an observation day in the tournament (there are 220 days in each game and 18 tournament games in a tournament). The input variables in this domain are observed prices for a specific tournament day. In addition, 4 time-delayed observations are included for each observed product and component (1, 2, 4 and 8 days delayed) to facilitate some anticipation of trends going forward. The datasets contain 16 regression targets, each target corresponds to the next day mean price (SCM1D) or mean price for 20-days in the future (SCM20D) for each product in the simulation. Days with no target values are excluded from the datasets (i.e. days with labels that are beyond the end of the game are excluded).

Appendix 2: Detailed experimental results

Base regressor exploration results

See Table 5.

Table 5 Detailed results for ST using RIDGE, TREE, SVR, BAG and SGB as base regressors

Dataset *Target*	ST-RIDGE	ST-TREE	ST-SVR	ST-BAG	ST-SGB
edm	0.871	0.915	0.860	0.742	**0.706**
dflow	0.977	1.092	0.961	0.815	**0.751**
dgap	0.764	0.737	0.760	0.669	**0.662**
sf1	1.130	1.127	**0.930**	1.135	1.148
c-class	**0.987**	1.045	1.001	1.017	1.026
m-class	**1.009**	1.345	1.084	1.096	1.111
x-class	1.394	0.991	**0.705**	1.293	1.307
sf2	1.485	1.010	**0.912**	1.149	1.223
c-class	0.953	0.973	**0.943**	0.980	0.980
m-class	1.136	1.006	**0.961**	1.075	1.112
x-class	2.365	1.050	**0.833**	1.393	1.578
jura	0.607	0.705	0.643	**0.589**	0.619
cd	0.715	0.854	0.727	**0.711**	0.750
co	0.626	0.616	0.679	**0.543**	0.550
cu	**0.480**	0.645	0.523	0.514	0.558
wq	0.955	0.963	0.959	**0.908**	0.922
25400	0.950	0.977	0.954	**0.925**	0.935
29600	0.987	1.011	0.996	0.987	**0.983**
30400	0.960	0.965	0.966	**0.945**	0.961
33400	0.950	0.974	0.956	**0.912**	0.929
17300	0.967	0.969	0.973	**0.902**	0.931
19400	0.909	0.890	0.910	**0.834**	0.840
34500	0.971	1.019	0.982	**0.969**	0.982
38100	0.957	0.963	0.956	**0.912**	0.924
49700	0.919	0.942	0.939	**0.795**	0.816
50390	0.966	0.966	0.966	**0.892**	0.899
55800	0.959	0.974	0.971	**0.924**	0.935
57500	0.961	0.969	0.962	**0.918**	0.924
59300	0.992	0.980	0.991	**0.947**	0.972
37880	0.917	0.888	0.904	**0.856**	0.878
enb	0.315	0.126	0.607	0.117	**0.114**
y1	0.293	0.062	0.587	**0.053**	0.059
y2	0.336	0.189	0.627	0.180	**0.168**

Table 5 continued

Dataset *Target*	ST-RIDGE	ST-TREE	ST-SVR	ST-BAG	ST-SGB
slump	0.679	0.827	0.686	0.688	**0.669**
slump	0.889	0.974	0.895	0.795	**0.744**
flow	0.774	0.857	0.755	0.742	**0.739**
compressi.	**0.373**	0.650	0.409	0.526	0.523
andro	0.842	0.599	1.109	0.602	**0.494**
1	0.801	0.663	1.038	0.515	**0.454**
2	0.799	0.333	0.863	0.340	**0.219**
3	1.079	0.592	1.386	0.588	**0.406**
4	1.079	0.522	1.458	0.530	**0.466**
5	**0.676**	0.704	0.985	0.809	0.686
6	**0.620**	0.783	0.921	0.827	0.731
osales	0.900	0.920	0.847	**0.748**	0.807
m1	0.919	0.829	0.818	**0.653**	0.944
m2	0.916	0.868	0.824	**0.754**	0.818
m3	0.884	0.916	0.824	**0.786**	0.800
m4	0.909	0.812	0.818	**0.689**	0.732
m5	0.991	0.917	0.848	**0.736**	0.790
m6	0.819	0.965	0.814	**0.696**	0.711
m7	0.883	0.960	0.819	**0.743**	0.825
m8	0.911	0.870	0.841	**0.764**	0.782
m9	0.872	1.205	0.884	**0.812**	0.915
m10	0.931	0.881	0.873	**0.773**	0.785
m11	0.845	0.841	0.872	0.749	**0.736**
m12	0.919	0.981	0.932	**0.821**	0.846
scpf	0.853	0.934	0.903	**0.837**	0.913
views	0.822	0.878	0.820	**0.815**	0.887
votes	0.752	0.813	0.843	**0.720**	0.754
comments	0.985	1.111	1.045	**0.976**	1.099
atp1d	0.412	0.473	0.432	**0.374**	0.395
allminpa	0.476	0.652	0.491	0.482	**0.470**
allminp0	**0.412**	0.567	0.427	0.430	0.468
adlminpa	0.419	0.468	0.433	**0.416**	0.428
acominpa	0.342	0.274	0.381	**0.242**	0.273
aflminpa	0.472	0.647	0.466	0.471	**0.464**
auaminpa	0.349	0.229	0.392	**0.200**	0.267

Table 5 continued

Dataset *Target*	ST-ridge	ST-tree	ST-svr	ST-bag	ST-sgb
atp7d	0.579	0.665	0.615	0.525	**0.517**
allminpa	0.730	0.950	0.764	0.641	**0.558**
allminp0	**0.575**	0.811	0.595	0.670	0.635
adlminpa	0.566	0.699	0.577	0.546	**0.533**
acominpa	0.437	0.355	0.509	**0.316**	0.321
aflminpa	0.717	0.821	0.730	**0.689**	0.745
auaminpa	0.448	0.353	0.514	**0.286**	0.313
oes97	**0.477**	0.709	0.579	0.525	0.644
58028	0.199	0.562	**0.183**	0.336	0.462
15014	**0.282**	0.510	0.292	0.358	0.554
32511	0.746	0.775	1.113	**0.717**	0.754
15017	0.322	0.454	**0.316**	0.365	0.380
98502	**0.602**	0.815	0.896	0.665	0.682
92965	**0.595**	0.961	0.766	0.648	0.766
32314	0.786	0.827	0.959	**0.614**	0.837
13008	0.228	0.505	**0.217**	0.339	0.517
21114	0.210	0.303	**0.209**	0.307	0.395
85110	**0.448**	0.754	0.770	0.567	0.669
27311	**0.573**	0.905	0.652	0.601	0.786
98902	**0.457**	0.850	0.461	0.536	0.689
65032	**0.470**	0.754	0.484	0.552	0.580
92998	**0.615**	0.828	0.617	0.644	0.701
27108	**0.544**	0.700	0.669	0.580	0.700
53905	**0.553**	0.839	0.665	0.569	0.825
oes10	**0.369**	0.621	0.427	0.420	0.569
513021	**0.368**	0.646	0.481	0.438	0.416
292071	**0.367**	0.599	0.413	0.385	0.404
392021	0.425	0.559	0.534	**0.412**	0.492
151131	**0.346**	0.655	0.363	0.430	0.649
151141	**0.355**	0.502	0.434	0.436	0.455
291069	**0.576**	0.971	0.646	0.616	0.876
119032	0.285	0.579	**0.280**	0.370	0.490
432011	**0.321**	0.589	0.426	0.392	0.487
419022	**0.522**	0.858	0.636	0.644	0.686
292037	**0.325**	0.459	0.376	0.335	0.412
519061	0.333	0.606	**0.331**	0.395	0.426
291051	**0.180**	0.347	0.192	0.265	0.404
172141	0.567	0.814	0.836	**0.494**	0.856
431011	0.150	0.456	**0.139**	0.269	0.616
291127	**0.386**	0.674	0.412	0.462	0.564
412021	0.397	0.615	**0.338**	0.377	0.878

Table 5 continued

Dataset *Target*	ST-RIDGE	ST-TREE	ST-SVR	ST-BAG	ST-SGB
rf1	0.541	0.121	0.414	**0.097**	0.230
chsi2	0.334	0.077	0.214	**0.043**	0.153
nasi2	1.951	**0.376**	1.301	0.432	0.653
eadm7	0.399	0.091	0.329	**0.055**	0.182
sclm7	0.600	0.128	0.367	**0.061**	0.222
clkm7	0.251	0.079	0.236	**0.049**	0.161
vali2	0.341	0.089	0.366	**0.057**	0.216
napm7	0.242	0.053	0.245	**0.040**	0.102
dldi4	0.211	0.073	0.253	**0.039**	0.151
rf2	0.469	0.121	0.414	**0.102**	0.237
chsi2	0.228	0.078	0.214	**0.044**	0.132
nasi2	1.996	**0.384**	1.296	0.465	0.780
eadm7	0.253	0.095	0.329	**0.055**	0.167
sclm7	0.294	0.105	0.367	**0.065**	0.186
clkm7	0.297	0.082	0.236	**0.050**	0.162
vali2	0.286	0.090	0.370	**0.056**	0.213
napm7	0.208	0.055	0.245	**0.041**	0.105
dldi4	0.190	0.078	0.253	**0.040**	0.153
scm1d	0.394	0.444	0.457	**0.348**	0.393
lbl	0.337	0.379	0.409	**0.310**	0.335
mtlp2	0.350	0.401	0.436	**0.323**	0.366
mtlp3	0.379	0.431	0.442	**0.333**	0.388
mtlp4	0.387	0.458	0.461	**0.345**	0.385
mtlp5	0.454	0.508	0.530	**0.377**	0.452
mtlp6	0.456	0.506	0.540	**0.376**	0.437
mtlp7	0.449	0.496	0.526	**0.370**	0.455
mtlp8	0.451	0.523	0.497	**0.377**	0.445
mtlp9	0.372	0.421	0.456	**0.341**	0.378
mtlp10	0.394	0.428	0.456	**0.352**	0.424
mtlp11	0.377	0.417	0.445	**0.342**	0.364
mtlp12	0.404	0.447	0.466	**0.366**	0.411
mtlp13	0.363	0.419	0.409	**0.331**	0.357
mtlp14	0.394	0.472	0.432	**0.369**	0.383
mtlp15	0.356	0.406	0.393	**0.330**	0.352
mtlp16	0.373	0.398	0.407	**0.330**	0.361

Table 5 continued

Dataset *Target*	ST-RIDGE	ST-TREE	ST-SVR	ST-BAG	ST-SGB
scm20d	0.646	0.627	0.763	**0.475**	0.620
lbl	0.561	0.562	0.678	**0.424**	0.542
mtlp2a	0.571	0.574	0.688	**0.425**	0.542
mtlp3a	0.586	0.572	0.683	**0.440**	0.564
mtlp4a	0.614	0.612	0.730	**0.455**	0.580
mtlp5a	0.707	0.681	0.846	**0.493**	0.697
mtlp6a	0.694	0.650	0.843	**0.493**	0.641
mtlp7a	0.691	0.644	0.833	**0.485**	0.624
mtlp8a	0.690	0.648	0.851	**0.500**	0.655
mtlp9a	0.644	0.604	0.737	**0.454**	0.599
mtlp10a	0.660	0.630	0.753	**0.478**	0.663
mtlp11a	0.666	0.667	0.769	**0.498**	0.632
mtlp12a	0.673	0.668	0.787	**0.511**	0.669
mtlp13a	0.651	0.634	0.751	**0.481**	0.611
mtlp14a	0.661	0.645	0.779	**0.501**	0.683
mtlp15a	0.633	0.613	0.727	**0.480**	0.608
mtlp16a	0.636	0.635	0.754	**0.488**	0.610

For each dataset, we first report the average $RRMSE$ over all targets ($aRRMSE$), and then the $RRMSE$ per target. In each row, the lowest error is typeset in bold

6.1 Multi-target regression results

See Table 6.

Table 6 Detailed results for all methods using BAG as base regressor in ST, SST, ERC and RLC (detailed results with additional base regressors can be found at http://users.auth.gr/espyromi/mtr/results.zip)

Dataset *Target*	ST	SST_{true}	SST_{train}	SST_{cv}	ERC_{true}	ERC_{train}	ERC_{cv}	MORF	RLC	TNR	Dirty
edm	0.742	0.747	0.743	0.740	0.743	0.742	0.741	**0.734**	0.735	0.851	0.830
dflow	0.815	0.824	0.817	0.812	0.818	0.815	0.814	**0.775**	0.801	0.932	0.900
dgap	0.669	0.671	0.669	**0.667**	0.669	0.669	0.668	0.692	0.669	0.769	0.759
sf1	1.135	**0.997**	1.127	1.068	1.050	1.132	1.089	1.282	1.163	1.112	1.115
c-class	1.017	0.991	1.037	1.001	1.000	1.020	1.007	1.035	1.019	0.974	**0.973**
m-class	1.096	**0.918**	1.137	1.005	0.992	1.112	1.033	1.212	1.130	0.998	1.016
x-class	1.293	**1.083**	1.207	1.198	1.158	1.263	1.226	1.601	1.341	1.365	1.356
sf2	1.149	0.980	**0.945**	1.055	1.053	1.087	1.088	1.425	1.228	1.475	1.372
c-class	0.980	0.968	0.975	0.964	0.973	0.977	0.964	0.996	0.985	0.948	**0.944**
m-class	1.075	**0.983**	0.994	0.992	1.022	1.018	1.030	1.160	1.080	1.118	1.103
x-class	1.393	0.988	**0.866**	1.210	1.165	1.266	1.270	2.119	1.620	2.360	2.069

Table 6 continued

Dataset *Target*	ST	SST$_{true}$	SST$_{train}$	SST$_{cv}$	ERC$_{true}$	ERC$_{train}$	ERC$_{cv}$	MORF	RLC	TNR	Dirty
jura	**0.589**	0.594	0.592	0.591	0.591	0.590	0.590	0.597	0.596	0.608	0.610
cd	0.711	0.715	0.715	0.713	0.712	0.713	0.712	**0.694**	0.702	0.715	0.716
co	0.543	0.553	0.544	0.543	0.546	**0.542**	0.543	0.566	0.558	0.628	0.629
cu	0.514	0.514	0.517	0.515	0.514	0.515	0.514	0.530	0.530	**0.481**	0.485
wq	0.908	0.914	0.911	0.909	0.910	0.905	0.906	**0.899**	0.902	0.962	0.961
25400	0.925	0.931	0.930	0.928	0.930	**0.921**	0.927	0.924	0.922	0.952	0.956
29600	0.987	0.988	0.986	0.984	0.985	0.983	0.983	**0.976**	0.979	0.994	0.995
30400	0.945	0.945	0.945	0.951	0.944	0.944	0.946	0.942	**0.937**	0.968	0.964
33400	0.912	0.915	0.904	0.902	0.912	0.904	0.902	**0.893**	0.904	0.959	0.957
17300	0.902	0.913	0.921	0.914	0.906	0.910	0.908	**0.895**	0.903	0.974	0.973
19400	0.834	0.839	0.829	0.829	0.835	**0.827**	0.829	0.828	0.832	0.912	0.907
34500	0.969	0.965	0.961	0.963	0.965	0.958	0.957	0.959	**0.957**	0.981	0.976
38100	0.912	0.913	0.908	0.908	0.912	0.906	0.911	0.907	**0.904**	0.967	0.967
49700	0.795	0.809	0.815	0.808	0.799	0.796	0.796	**0.793**	0.793	0.927	0.936
50390	0.892	0.899	0.901	0.899	0.892	0.892	0.892	0.892	**0.884**	0.988	0.961
55800	0.924	0.934	0.931	0.929	0.926	0.922	0.924	**0.903**	0.916	0.963	0.973
57500	0.918	0.920	0.917	0.919	0.918	0.915	0.914	**0.896**	0.907	0.965	0.964
59300	0.947	0.965	0.957	0.951	0.954	0.946	0.947	**0.931**	0.941	0.995	0.994
37880	0.856	0.860	0.848	**0.847**	0.857	0.852	0.849	0.851	0.851	0.923	0.927
enb	0.117	0.145	0.123	0.121	0.125	0.117	**0.114**	0.121	0.120	0.316	0.319
y1	0.053	0.088	0.059	0.063	0.064	0.053	0.053	0.060	**0.053**	0.295	0.297
y2	0.180	0.201	0.186	0.178	0.187	0.181	**0.174**	0.182	0.188	0.337	0.341
slump	0.688	0.722	**0.666**	0.695	0.701	0.669	0.689	0.694	0.690	0.681	0.684
slump	0.795	0.857	**0.742**	0.802	0.817	0.757	0.796	0.775	0.792	0.900	0.900
flow	0.742	0.767	0.743	0.764	0.752	**0.732**	0.749	0.733	0.740	0.769	0.769
compressi.	0.526	0.540	0.514	0.521	0.533	0.519	0.522	0.573	0.539	**0.372**	0.384
andro	0.602	0.603	0.540	0.579	0.596	0.538	0.567	**0.510**	0.570	0.803	0.889
1	0.515	0.584	0.487	0.507	0.537	0.485	0.504	**0.436**	0.452	0.860	0.792
2	0.340	0.349	0.340	0.336	0.344	0.339	0.332	0.403	**0.296**	1.000	0.934
3	0.588	0.497	**0.404**	0.518	0.540	0.435	0.475	0.499	0.594	0.833	1.009
4	0.530	0.551	0.500	0.579	0.521	0.483	0.521	**0.467**	0.587	0.837	1.052
5	0.809	0.843	0.842	0.800	0.819	0.758	0.788	**0.632**	0.745	0.657	0.761
6	0.827	0.795	0.667	0.736	0.816	0.730	0.781	**0.622**	0.747	0.629	0.782
osales	0.748	0.751	0.709	0.726	0.728	**0.699**	0.713	0.753	0.741	1.628	1.507
m1	0.653	0.695	0.604	0.596	0.674	**0.590**	0.596	0.676	0.657	1.881	1.737
m2	0.754	0.740	0.696	0.756	0.733	**0.675**	0.732	0.719	0.750	1.551	1.643
m3	0.786	0.809	0.779	0.825	**0.760**	0.772	0.773	0.778	0.772	1.639	1.630
m4	0.689	0.728	**0.636**	0.660	0.696	0.643	0.659	0.736	0.676	1.598	1.640
m5	0.736	0.668	0.669	0.652	0.692	0.653	**0.647**	0.738	0.703	2.319	1.810
m6	0.696	0.763	0.725	0.735	0.704	**0.671**	0.699	0.753	0.710	1.562	1.555
m7	0.743	0.726	0.682	0.691	0.712	**0.673**	0.679	0.768	0.738	1.572	1.322

Table 6 continued

Dataset *Target*	ST	SST_{true}	SST_{train}	SST_{cv}	ERC_{true}	ERC_{train}	ERC_{cv}	MORF	RLC	TNR	Dirty
m8	0.764	0.760	0.729	**0.726**	0.746	0.730	0.733	0.789	0.759	1.359	1.293
m9	0.812	0.767	0.755	0.803	0.747	**0.741**	0.778	0.746	0.793	1.401	1.394
m10	0.773	0.763	0.729	0.733	0.738	**0.726**	0.732	0.770	0.776	1.905	1.471
m11	0.749	0.756	**0.708**	0.734	0.737	0.710	0.729	0.760	0.736	1.278	1.202
m12	0.821	0.843	0.799	0.801	**0.797**	0.798	0.801	0.806	0.818	1.471	1.384
scpf	0.837	0.830	0.855	0.831	**0.812**	0.821	0.830	0.833	0.835	0.899	0.877
views	0.815	0.795	0.811	0.815	**0.787**	0.789	0.810	0.808	0.814	0.855	0.857
votes	0.720	0.750	0.759	0.718	0.718	0.724	0.719	**0.704**	0.724	0.825	0.752
comments	0.976	0.945	0.996	0.960	**0.931**	0.950	0.961	0.988	0.966	1.015	1.020
atp1d	0.374	0.376	0.372	0.372	0.371	**0.367**	0.372	0.422	0.378	0.595	0.552
allminpa	0.482	0.506	0.485	0.482	0.485	0.481	0.482	**0.474**	0.475	0.687	0.644
allminp0	0.430	0.428	0.420	0.422	0.426	**0.416**	0.428	0.436	0.434	0.687	0.628
adlminpa	0.416	**0.397**	0.405	0.410	0.398	0.402	0.412	0.424	0.419	0.564	0.519
acominpa	0.242	0.245	**0.233**	0.242	0.242	0.234	0.239	0.356	0.248	0.481	0.441
aflminpa	0.471	0.468	0.480	0.473	0.470	0.470	0.473	0.487	**0.461**	0.654	0.627
auaminpa	**0.200**	0.210	0.207	0.202	0.205	0.201	0.200	0.356	0.232	0.497	0.453
atp7d	0.525	0.561	0.514	**0.507**	0.534	0.509	0.512	0.551	0.529	0.786	0.668
allminpa	0.641	0.765	**0.619**	0.629	0.696	0.619	0.626	0.636	0.673	0.828	0.761
allminp0	0.670	0.698	0.672	0.653	0.675	0.666	0.660	**0.602**	0.644	0.827	0.738
adlminpa	0.546	0.588	0.533	0.537	0.553	0.530	0.542	**0.524**	0.555	0.784	0.624
acominpa	0.316	0.301	**0.269**	0.278	0.294	0.283	0.291	0.437	0.328	0.705	0.561
aflminpa	0.689	0.727	0.728	0.700	0.707	0.692	0.694	0.674	**0.669**	0.852	0.745
auaminpa	0.286	0.288	0.264	**0.246**	0.281	0.267	0.262	0.432	0.304	0.721	0.577
oes97	0.525	0.526	0.526	0.524	0.525	0.525	0.524	0.549	**0.523**	0.818	1.049
58028	0.336	0.337	0.335	0.337	0.337	0.335	0.336	**0.265**	0.312	0.354	0.389
15014	0.358	0.359	0.365	**0.352**	0.359	0.361	0.352	0.465	0.367	0.446	0.560
32511	0.717	0.715	0.715	**0.714**	0.716	0.716	0.714	0.752	0.733	1.398	2.077
15017	0.365	**0.365**	0.370	0.368	0.365	0.367	0.366	0.403	0.373	0.505	0.743
98502	0.665	0.675	0.666	0.665	0.671	0.666	0.665	0.715	**0.659**	1.219	1.653
92965	0.648	0.651	0.648	0.647	0.649	0.647	0.647	0.698	**0.640**	0.970	1.626
32314	0.614	**0.611**	0.615	0.617	0.613	0.614	0.615	0.704	0.633	0.915	1.137
13008	0.339	0.339	0.340	0.337	0.339	0.340	0.338	**0.272**	0.321	0.318	0.460
21114	0.307	0.307	0.307	0.308	0.307	0.307	0.307	0.291	**0.283**	0.419	0.458
85110	0.567	0.567	0.575	0.568	0.567	0.569	0.566	0.636	**0.566**	0.902	1.186
27311	0.601	0.598	0.595	0.595	0.599	0.598	0.598	**0.566**	0.584	0.952	1.108
98902	0.536	0.537	0.537	0.536	0.536	0.536	0.535	0.543	**0.513**	0.704	0.743
65032	0.552	0.552	0.554	0.550	0.552	0.553	0.551	**0.535**	0.542	0.877	0.956
92998	0.644	0.653	**0.642**	0.644	0.647	0.643	0.642	0.703	0.659	1.198	1.252
27108	0.580	0.582	**0.576**	0.577	0.581	0.578	0.577	0.636	0.596	0.842	1.256
53905	**0.569**	0.572	0.574	0.572	0.570	0.571	0.570	0.602	0.591	1.064	1.173

Table 6 continued

Dataset *Target*	ST	SST_{true}	SST_{train}	SST_{cv}	ERC_{true}	ERC_{train}	ERC_{cv}	MORF	RLC	TNR	Dirty
oes10	0.420	0.421	0.420	0.421	0.420	0.420	0.420	0.452	**0.419**	0.532	0.664
513021	0.438	**0.436**	0.438	0.437	0.436	0.438	0.438	0.453	0.443	0.587	0.824
292071	0.385	0.387	0.383	**0.382**	0.387	0.383	0.383	0.393	0.388	0.548	0.733
392021	0.412	0.413	0.413	0.414	0.413	0.412	0.412	**0.397**	0.415	0.605	0.807
151131	0.430	0.431	0.431	0.431	0.431	0.431	0.430	0.457	**0.423**	0.511	0.515
151141	**0.436**	0.440	0.439	0.441	0.438	0.437	0.438	0.496	0.446	0.508	0.770
291069	0.616	0.618	0.617	0.616	0.617	0.617	0.616	0.650	**0.589**	0.834	0.874
119032	0.370	0.370	0.369	0.371	0.370	0.368	0.370	0.402	0.384	**0.332**	0.509
432011	0.392	0.393	0.392	0.392	0.392	0.392	0.392	0.397	**0.391**	0.471	0.661
419022	0.644	0.644	0.644	0.644	0.644	0.645	0.644	0.654	**0.610**	0.779	0.914
292037	**0.335**	0.336	0.337	0.337	0.335	0.336	0.335	0.382	0.352	0.432	0.606
519061	**0.395**	0.396	0.397	0.397	0.395	0.396	0.396	0.557	0.404	0.558	0.608
291051	0.265	0.266	0.267	0.266	0.265	0.265	0.264	0.266	0.262	**0.220**	0.297
172141	**0.494**	0.495	0.496	0.496	0.495	0.496	0.494	0.593	0.520	1.067	1.080
431011	0.269	0.268	0.267	0.271	0.268	0.268	0.269	0.227	0.240	0.166	**0.165**
291127	0.462	0.460	0.453	0.456	0.461	0.456	0.458	**0.442**	0.456	0.449	0.794
412021	0.377	**0.375**	0.378	0.379	0.376	0.377	0.378	0.461	0.378	0.437	0.474
rf1	0.097	0.113	0.094	0.094	0.101	0.091	**0.091**	0.123	0.121	0.983	0.676
chsi2	0.043	0.069	0.047	0.046	0.050	0.035	0.034	0.035	**0.033**	0.797	0.387
nasi2	0.432	0.498	0.431	0.431	0.454	0.430	**0.430**	0.650	0.684	1.946	2.610
eadm7	0.055	0.050	0.043	0.040	0.047	0.041	0.040	**0.039**	0.042	1.019	0.524
sclm7	0.061	0.069	0.049	0.048	0.060	0.048	**0.047**	0.068	0.048	1.503	0.834
clkm7	0.049	0.048	0.041	0.041	0.045	0.040	0.041	**0.031**	0.041	0.587	0.256
vali2	0.057	0.074	0.056	0.057	0.069	0.055	0.055	**0.037**	0.046	0.571	0.320
napm7	0.040	0.051	0.044	0.047	0.045	0.039	0.041	0.097	**0.038**	0.909	0.270
dldi4	0.039	0.046	0.038	0.038	0.041	0.037	0.038	**0.029**	0.031	0.534	0.208
rf2	0.102	0.123	0.100	0.097	0.109	0.096	**0.095**	0.148	0.130	1.103	0.586
chsi2	0.044	0.077	0.042	0.046	0.059	0.035	**0.034**	0.066	0.038	0.737	0.258
nasi2	0.465	0.534	0.470	**0.456**	0.482	0.461	0.461	0.587	0.747	3.143	2.597
eadm7	0.055	0.055	0.047	**0.040**	0.052	0.043	0.041	0.071	0.046	0.737	0.319
sclm7	0.065	0.089	0.056	0.050	0.074	0.053	**0.049**	0.119	0.051	0.970	0.370
clkm7	0.050	0.049	**0.041**	0.041	0.048	0.042	0.042	0.054	0.042	0.891	0.420
vali2	0.056	0.074	0.056	0.054	0.069	0.054	0.053	0.067	**0.047**	0.956	0.270
napm7	0.041	0.055	0.051	0.049	0.047	0.044	0.043	0.166	**0.039**	0.617	0.273
dldi4	0.040	0.049	0.039	0.039	0.042	0.039	0.039	0.053	**0.032**	0.770	0.180
scm1d	0.348	0.360	0.340	0.336	0.353	0.332	**0.330**	0.352	0.345	0.437	0.399
lbl	0.310	0.312	0.300	0.296	0.310	0.294	**0.294**	0.308	0.306	0.378	0.341
mtlp2	0.323	0.334	0.317	0.313	0.329	0.309	**0.308**	0.327	0.317	0.408	0.353
mtlp3	0.333	0.351	0.325	0.322	0.342	0.319	**0.315**	0.328	0.327	0.429	0.384
mtlp4	0.345	0.362	0.338	0.334	0.359	0.330	**0.325**	0.343	0.342	0.442	0.393

Table 6 continued

Dataset *Target*	ST	SST$_{true}$	SST$_{train}$	SST$_{cv}$	ERC$_{true}$	ERC$_{train}$	ERC$_{cv}$	MORF	RLC	TNR	Dirty
mtlp5	0.377	0.404	0.366	0.365	0.395	0.357	**0.349**	0.391	0.377	0.502	0.461
mtlp6	0.376	0.392	0.357	0.350	0.381	0.356	**0.347**	0.399	0.382	0.509	0.462
mtlp7	0.370	0.380	0.350	0.343	0.376	0.346	**0.338**	0.383	0.369	0.497	0.459
mtlp8	0.377	0.393	0.362	0.356	0.390	0.353	**0.345**	0.391	0.378	0.504	0.459
mtlp9	0.341	0.349	0.333	0.330	0.344	**0.323**	0.325	0.340	0.332	0.411	0.376
mtlp10	0.352	0.359	0.350	0.344	0.351	**0.339**	0.341	0.348	0.348	0.434	0.398
mtlp11	0.342	0.348	0.331	0.329	0.342	**0.327**	0.329	0.346	0.336	0.410	0.384
mtlp12	0.366	0.366	0.355	0.355	0.362	**0.350**	0.354	0.363	0.364	0.441	0.412
mtlp13	0.331	0.348	0.334	0.326	0.337	0.323	**0.322**	0.345	0.326	0.399	0.367
mtlp14	0.369	0.374	0.363	0.360	0.367	**0.356**	0.358	0.361	0.358	0.438	0.398
mtlp15	0.330	0.337	0.321	0.319	0.327	**0.314**	0.315	0.330	0.322	0.389	0.361
mtlp16	0.330	0.346	0.333	0.329	0.334	0.322	**0.322**	0.335	0.332	0.405	0.380
scm20d	0.475	0.493	0.431	0.413	0.497	0.415	**0.394**	0.443	0.472	0.655	0.658
lbl	0.424	0.441	0.386	0.368	0.448	0.372	**0.356**	0.393	0.423	0.569	0.561
mtlp2a	0.425	0.442	0.384	0.365	0.455	0.369	**0.352**	0.402	0.429	0.581	0.573
mtlp3a	0.440	0.453	0.402	0.387	0.464	0.385	**0.363**	0.415	0.437	0.593	0.587
mtlp4a	0.455	0.471	0.410	0.396	0.485	0.397	**0.374**	0.426	0.453	0.622	0.618
mtlp5a	0.493	0.535	0.459	0.441	0.532	0.432	**0.413**	0.467	0.489	0.711	0.737
mtlp6a	0.493	0.512	0.450	0.430	0.504	0.438	**0.424**	0.466	0.488	0.701	0.728
mtlp7a	0.485	0.499	0.440	0.422	0.502	0.422	**0.404**	0.450	0.482	0.698	0.718
mtlp8a	0.500	0.504	0.447	0.431	0.513	0.426	**0.407**	0.454	0.493	0.696	0.725
mtlp9a	0.454	0.481	0.421	0.402	0.489	0.404	**0.382**	0.437	0.458	0.650	0.647
mtlp10a	0.478	0.497	0.436	**0.413**	0.498	0.435	0.418	0.454	0.489	0.668	0.666
mtlp11a	0.498	0.502	0.428	0.407	0.505	0.426	**0.402**	0.449	0.490	0.671	0.670
mtlp12a	0.511	0.525	0.458	0.444	0.520	0.449	**0.429**	0.473	0.511	0.680	0.681
mtlp13a	0.481	0.515	0.453	0.431	0.516	0.426	**0.400**	0.450	0.471	0.663	0.658
mtlp14a	0.501	0.517	0.455	0.440	0.520	0.436	**0.411**	0.465	0.492	0.675	0.668
mtlp15a	0.480	0.496	0.432	0.411	0.503	0.409	**0.384**	0.439	0.472	0.648	0.638
mtlp16a	0.488	0.496	0.434	0.414	0.504	0.411	**0.386**	0.445	0.481	0.647	0.646

For each dataset, we first report the average $RRMSE$ over all targets ($aRRMSE$), and then the $RRMSE$ per target. In each row, the lowest error is typeset in bold

References

Aho, T., Zenko, B., Dzeroski, S., & Elomaa, T. (2012). Multi-target regression with rule ensembles. *Journal of Machine Learning Research*, *13*, 2367–2407.

Álvarez, M. A., & Lawrence, N. D. (2011). Computationally efficient convolved multiple output gaussian processes. *Journal of Machine Learning Research*, *12*, 1459–1500.

Alvarez, M. A., Rosasco, L., & Lawrence, N. D. (2011). Kernels for vector-valued functions: A review. arXiv preprint arXiv:1106.6251.

Ando, R. K., & Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, *6*, 1817–1853.

Argyriou, A., Evgeniou, T., & Pontil, M. (2006). Multi-task feature learning. In *Advances in neural information processing systems 19, Proceedings of the twentieth annual conference on neural information processing systems*, Vancouver, British Columbia, Canada, December 4–7, 2006, pp. 41–48.

Argyriou, A., Evgeniou, T., & Pontil, M. (2008). Convex multi-task feature learning. *Machine Learning*, *73*(3), 243–272.

Balasubramanian, K., & Lebanon, G. (2012). The landmark selection method for multiple output prediction. In *Proceedings of the 29th international conference on machine learning*, ICML 2012, Edinburgh, Scotland, UK, June 26–July 1, 2012.

Baxter, J. (1995). Learning internal representations. In *Proceedings of the eigth annual conference on computational learning theory*, COLT 1995, Santa Cruz, California, USA, July 5–8, 1995, pp. 311–320.

Blockeel, H., Raedt, L. D., & Ramon, J. (1998). Top-down induction of clustering trees. In *Proceedings of the fifteenth international conference on machine learning* (ICML 1998), Madison, Wisconsin, USA, July 24–27, 1998, pp. 55–63.

Blockeel, H., Dzeroski, S., & Grbovic, J. (1999). Simultaneous prediction of mulriple chemical parameters of river water quality with TILDE. In *Proceedings of third European conference in principles of data mining and knowledge discovery* PKDD'99, Prague, Czech Republic, September 15–18, 1999, pp. 32–40.

Bonilla, E. V., Chai, K. M. A., & Williams, C. K. I. (2007). Multi-task gaussian process prediction. In *Advances in neural information processing systems 20, proceedings of the twenty-first annual conference on neural information processing systems*, Vancouver, British Columbia, Canada, December 3–6, 2007, pp. 153–160.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Breiman, L., & Friedman, J. H. (1997). Predicting multivariate responses in multiple linear regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *59*(1), 3–54.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Belmont, CA: Wadsworth.

Caruana, R. (1994). Learning many related tasks at the same time with backpropagation. In *Advances in neural information processing systems 7*, [NIPS Conference, Denver, Colorado, USA, 1994], pp. 657–664.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*(1), 41–75.

Chen, J., Tang, L., Liu, J., & Ye, J. (2009). A convex formulation for learning shared structures from multiple tasks. In *Proceedings of the 26th annual international conference on machine learning*, ICML 2009, Montreal, Quebec, Canada, June 14–18, 2009, pp. 137–144.

Chen, J., Liu, J., & Ye, J. (2010a). Learning incoherent sparse and low-rank patterns from multiple tasks. In *Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining*, Washington, DC, USA, July 25–28, 2010, pp. 1179–1188.

Chen, X., Kim, S., Lin, Q., Carbonell, J. G., & Xing, E. P. (2010b). Graph-structured multi-task regression and an efficient optimization method for general fused lasso. arXiv preprint arXiv:1005.3579.

Cheng, W., & Hüllermeier, E. (2009). Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, *76*(2–3), 211–225.

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the twenty-fifth international conference on machine learning*, (ICML 2008), Helsinki, Finland, June 5–9, 2008, pp. 160–167.

Dembczynski, K., Cheng, W., & Hüllermeier, E. (2010). Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th international conference on machine learning* (ICML-10), June 21-24, 2010, Haifa, Israel, pp. 279–286.

Dembczynski, K., Waegeman, W., Cheng, W., & Hüllermeier, E. (2012). On label dependence and loss minimization in multi-label classification. *Machine Learning*, *88*(1–2), 5–45.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A. J., & Vapnik, V. (1996). Support vector regression machines. In *Advances in neural information processing systems 9*, NIPS, Denver, CO, USA, December 2–5, 1996, pp. 155–161.

Dzeroski, S., Demsar, D., & Grbovic, J. (2000). Predicting chemical parameters of river water quality from bioindicator data. *Applied Intelligence*, *13*(1), 7–17.

Evgeniou, T., & Pontil, M. (2004). Regularized multi-task learning. In *Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining*, Seattle, Washington, USA, August 22–25, 2004, pp. 109–117.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning*. Berlin: Springer series in statistics Springer.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, *38*(4), 367–378.

Ghosn, J., & Bengio, Y. (1996). Multi-task learning for stock selection. In *Advances in neural information processing systems 9*, NIPS, Denver, CO, USA, December 2–5, 1996, pp. 946–952.

Godbole, S., & Sarawagi, S. (2004). Discriminative methods for multi-labeled classification. In *Proceedings of 8th Pacific-Asia conference on advances in knowledge discovery and data mining*, PAKDD 2004, Sydney, Australia, May 26–28, 2004, pp. 22–30.

Goovaerts, P. (1997). *Geostatistics for natural resources evaluation*. Oxford: Oxford University Press.

Groves, W., & Gini, M. L. (2011). Improving prediction in TAC SCM by integrating multivariate and temporal aspects via PLS regression. In *Agent-mediated electronic commerce. Designing trading strategies and mechanisms for electronic markets - AMEC 2011*, Taipei, Taiwan, May 2, 2011, and TADA 2011, Barcelona, Spain, July 17, 2011, Revised Selected Papers, pp. 28–43.

Groves, W., & Gini, M. L. (2015). On optimizing airline ticket purchase timing. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *7*(1), 3.

Hatzikos, E. V., Tsoumakas, G., Tzanis, G., Bassiliades, N., & Vlahavas, I. P. (2008). An empirical study on sea water quality prediction. *Knowledge-Based Systems*, *21*(6), 471–478.

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *12*(1), 55–67.

Hsu, D., Kakade, S., Langford, J., & Zhang, T. (2009). Multi-label prediction via compressed sensing. In *Advances in neural information processing systems 22: 23rd annual conference on neural information processing systems 2009. Proceedings of a meeting held* 7–10 December 2009, Vancouver, British Columbia, Canada, pp. 772–780.

Iman, R. L., & Davenport, J. M. (1980). Approximations of the critical region of the friedman statistic. *Communications in Statistics-Theory and Methods*, *9*(6), 571–595.

Izenman, A. J. (1975). Reduced-rank regression for the multivariate linear model. *Journal of Multivariate Analysis*, *5*(2), 248–264.

Izenman, A. J. (2008). *Modern multivariate statistical techniques: Regression, classification, and manifold learning*. New York: Springer.

Jacob, L., Bach, F. R., & Vert, J. (2008). Clustered multi-task learning: A convex formulation. In *Advances in neural information processing systems 21. Proceedings of the twenty-second annual conference on neural information processing systems*, Vancouver, British Columbia, Canada, December 8–11, 2008, pp 745–752.

Jalali, A., Ravikumar, P. D., Sanghavi, S., & Ruan, C. (2010) A dirty model for multi-task learning. In *Advances in neural information processing systems 23: 24th annual conference on neural information processing systems 2010. Proceedings of a meeting* held 6–9 December 2010, Vancouver, British Columbia, Canada, pp. 964–972.

Jalali, A., Ravikumar, P. D., & Sanghavi, S. (2013). A dirty model for multiple sparse regression. *IEEE Transactions on Information Theory*, *59*(12), 7947–7968.

Ji, S., & Ye, J. (2009). An accelerated gradient method for trace norm minimization. In *Proceedings of the 26th annual international conference on machine learning*, ICML 2009, Montreal, Quebec, Canada, June 14–18, 2009, pp. 457–464.

Kaggle. (2012). Kaggle competition: Online product sales. https://www.kaggle.com/c/online-sales

Kaggle. (2013). Kaggle competition: See click predict fix. https://www.kaggle.com/c/see-click-predict-fix

Karalic, A., & Bratko, I. (1997). First order regression. *Machine Learning*, *26*(2–3), 147–176.

Kim, S., & Xing, E.P. (2010). Tree-guided group lasso for multi-task regression with structured sparsity. In *Proceedings of the 27th international conference on machine learning* (ICML-10), June 21–24, 2010, Haifa, Israel, pp. 543–550.

Kocev, D., Vens, C., Struyf, J., & Dzeroski, S. (2007). Ensembles of multi-objective decision trees. In *Proceedings of 18th European conference on machine learning: ECML 2007*, Warsaw, Poland, September 17–21, 2007, pp. 624–631.

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.

Kumar, A., Vembu, S., Menon, A. K., & Elkan, C. (2012). Learning and inference in probabilistic classifier chains with beam search. In *Proceedings of European conference on machine learning and knowledge discovery in databases, Part I*, ECML PKDD 2012, Bristol, UK, September 24–28, 2012. pp. 665–680.

Lichman, M. (2013). UCI machine learning repository. http://archive.ics.uci.edu/ml

Luaces, O., Díez, J., Barranquero, J., del Coz, J. J., & Bahamonde, A. (2012). Binary relevance efficacy for multilabel classification. *Progress in Artificial Intelligence*, *1*(4), 303–313.

Montañés, E., Quevedo, J. R., & del Coz, J. J. (2011). Aggregating independent and dependent models to learn multi-label classifiers. In *Proceedings of European conference on machine learning and knowledge discovery in databases, Part II*, ECML PKDD 2011, Athens, Greece, September 5–9, 2011, pp. 484–500.

Munson, M. A., & Caruana, R. (2009). On feature selection, bias-variance, and bagging. In *Proceedings of European conference on machine learning and knowledge discovery in databases, Part II*, ECML PKDD 2009, Bled, Slovenia, September 7–11, 2009, pp. 144–159.

Obozinski, G., Taskar, B., & Jordan, M. I. (2010). Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, *20*(2), 231–252.

Pardoe, D., & Stone, P. (2008). The 2007 TAC SCM prediction challenge. In *AAAI 2008 workshop on trading agent design and analysis*.

Pratt, L. Y. (1992). Discriminability-based transfer between neural networks. In *Advances in neural information processing systems 5*, [NIPS Conference, Denver, Colorado, USA, November 30–December 3, 1992], pp. 204–211.

Read, J., & Hollmén, J. (2014). A deep interpretation of classifier chains. In *Proceedings of 13th international symposium on advances in intelligent data analysis XIII*, IDA 2014, Leuven, Belgium, October 30–November 1, 2014, pp. 251–262.

Read, J., & Hollmén, J. (2015). Multi-label classification using labels as hidden nodes. arXiv preprint arXiv:1503.09022.

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning*, *85*(3), 333–359.

Read, J., Martino, L., & Luengo, D. (2014). Efficient monte carlo methods for multi-dimensional learning with classifier chains. *Pattern Recognition*, *47*(3), 1535–1546.

Senge, R., del Coz, J. J., & Hüllermeier, E. (2013a) On the problem of error propagation in classifier chains for multi-label classification. In *Proceedings of the 36th annual conference of the German classification society*.

Senge, R., del Coz, J. J., & Hüllermeier, E. (2013b) Rectifying classifier chains for multi-label classification. In *LWA 2013. Lernen, Wissen & Adaptivität, workshop proceedings Bamberg*, 7–9 October 2013, pp. 151–158.

Spyromitros-Xioufis, E., Tsoumakas, G., Groves, W., & Vlahavas, I. (2012). Multi-label classification methods for multi-target regression. ArXiv e-prints arXiv:1211.6581v1.

Su, J., & Zhang, H. (2006) A fast decision tree learning algorithm. In *Proceedings, the twenty-first national conference on artificial intelligence and the eighteenth innovative applications of artificial intelligence conference*, July 16–20, 2006, Boston, Massachusetts, USA, pp 500–505

Tai, F., & Lin, H. (2012). Multilabel classification with principal label space transformation. *Neural Computation*, *24*(9), 2508–2542.

Teh, Y. W., Seeger, M., & Jordan, M. I. (2005). Semiparametric latent factor models. In *Proceedings of the tenth international workshop on artificial intelligence and statistics*, AISTATS 2005, Bridgetown, Barbados, January 6–8, 2005.

Tsanas, A., & Xifara, A. (2012). Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings*, *49*, 560–567.

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2010). Mining multi-label data. In O. Maimon & L. Rokach (Eds.), *Data mining and knowledge discovery handbook* (2nd ed., pp. 667–685). Boston, MA: Springer.

Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., & Vlahavas, I. (2011). Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, *12*, 2411–2414.

Tsoumakas, G., Xioufis, E.S., Vrekou, A., & Vlahavas, I. P. (2014). Multi-target regression via random linear target combinations. In *Proceedings of European conference on machine learning and knowledge discovery in databases, Part III*, ECML PKDD 2014, Nancy, France, September 15–19, 2014, pp. 225–240.

Van Der Merwe, A., & Zidek, J. (1980). Multivariate regression analysis and canonical variates. *Canadian Journal of Statistics*, *8*(1), 27–39.

Weston, J., Chapelle, O., Elisseeff, A., Schölkopf, B., & Vapnik, V. (2002). Kernel dependency estimation. In *Advances in neural information processing systems 15* [Neural Information Processing Systems, NIPS 2002, December 9–14, 2002, Vancouver, British Columbia, Canada], pp. 873–880.

Wold, H. (1985). Partial least squares. In S. Kotz & N. L. Johnson (Eds.), *Encyclopedia of statistical sciences* (Vol. 6, pp. 581–591). New York: Wiley.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*(2), 241–259.

Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural Computation*, *8*(7), 1341–1390.

Wolpert, D. H. (2002). The supervised learning no-free-lunch theorems. In R. Roy, M. Köppen, S. Ovaska, T. Furuhashi, & F. Hoffmann (Eds.), *Soft computing and industry: Recent applications* (pp. 25–42). London: Springer.

Yeh, I. C. (2007). Modeling slump flow of concrete using second-order regressions and artificial neural networks. *Cement and Concrete Composites*, *29*(6), 474–480.

Zhang, M., & Zhou, Z. (2014). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, *26*(8), 1819–1837.

Zhang, Y., & Schneider, J. G. (2011). Multi-label output codes using canonical correlation analysis. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, AISTATS 2011, Fort Lauderdale, USA, April 11–13, 2011, pp. 873–882.

Zhang, Y., & Schneider, J. G. (2012). Maximum margin output coding. In *Proceedings of the 29th international conference on machine learning*, ICML 2012, Edinburgh, Scotland, UK, June 26–July 1, 2012.

Zhou, J., Chen, J., & Ye, J. (2011a). Clustered multi-task learning via alternating structure optimization. In *Advances in neural information processing systems 24: 25th annual conference on neural information processing systems 2011. Proceedings of a meeting* held 12–14 December 2011, Granada, Spain, pp. 702–710.

Zhou, J., Chen, J., & Ye, J. (2011b). *Malsar: Multi-task learning via structural regularization*. Tempe: Arizona State University.

Zhou, J., Chen, J., & Ye, J. (2012). *Multi-task learning: Theory, algorithms, and applications*. https://www.siam.org/meetings/sdm12/zhou_chen_ye.pdf