MULTI-OBJECTIVE APPROACHES FOR MULTI-TARGET LEARNING

by

Esra Adıyeke

B.S., Mathematical Engineering, Yıldız Technical University, 2011

M.S., Industrial Engineering, Bahçeşehir University, 2014

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Industrial Engineering

Boğaziçi University

2020

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor Assist. Prof. Mustafa G. Baydoğan for his guidance, support and contribution.

I am grateful for the periodical feed-backs, suggestions and comments given by my thesis committee members, Assoc. Prof. Gönenç Yücel and Prof. Serpil Sayın. I would also like to thank Assist. Prof. Ethem Çanakoğlu and Prof. Z. Caner Taşkın for taking part in my thesis committee.

And finally, I would like to thank my family for their love and unconditional support throughout my life.

ABSTRACT

MULTI-OBJECTIVE APPROACHES FOR MULTI-TARGET LEARNING

Multi-target datasets (MTD) require simultaneous prediction of several variables hence they are considered to be more challenging in terms of predictive tasks compared to single-target datasets. Mining of MTD requires handling of several problems. To exemplify, scale inconsistencies are widely encountered in the targets. Most of the existing approaches resolve this issue by transforming the targets to the same scale, yet those operations may change the statistical properties of the dataset. Besides, features' scale inconsistencies cause problems in semi-supervised learning (SSL) applications since distance-based calculations are required therein. Another issue with MTD is, to explore alternative ways of including the target relations in learning applications. In this thesis, I develop supervised learning (SL), SSL and feature ranking (FR) models for MTD to deal with aforementioned problems. Benefiting from multi-objective optimization concepts, I aim to propose learning strategies that are robust to the type of the variables processed and utilize the target relations at the same time. Specifically, I propose a multi-objective extension for standard decision trees and a selective classifier chaining strategy for SL tasks. Experimental studies show that proposed models outperform their benchmark models. Besides, multi-objective trees extended to their semi-supervised version so that proposed form could result a competitive performance when the label information is not adequate. Performed experiments show a significant improvement of the proposed model over its benchmarks. In addition, since high-dimesionality and irrelevance in features reduce the effectiveness of a learning model, an embedded feature ranking (FR) procedure to semi-supervised trees is given to address this problem. Applications on several datasets show that, proposed FR procedure enhances the predictive performance compared to its benchmark approaches.

ÖZET

ÇOK DEĞİŞKENLİ ÖĞRENMEDE ÇOK HEDEFLİ YAKLAŞIMLAR

Çok hedefli veri kümeleri (ÇHVK), birkaç değişkenin eş zamanlı olarak tahminini gerektirir. Bu tip veri kümeleri birden fazla değişken için tahmin gerektirmesinden ötürü tek hedefli veri kümelerine kıyasla daha zorlayıcı olarak kabul edilirler. ÇHVK için olan öğrenme uygulamaları birtakım problemlerin ele alınmasını gerektirmektedir. Örneğin, hedeflerde ölçek tutarsızlıklarına yaygın olarak rastlanmaktadır. Genellikle bu durum dönüştürme işlemleriyle çözülmektedir ancak bu işlemler veri kümesinin istatistiksel özelliklerini değiştirebilmektedir. Özelliklerde gözlenen ölçek tutarsızlıkları, yarı denetimli öğrenme (YDÖ) uygulamalarında mesafeye dayalı hesaplamalara ihtiyaç duyulduğundan sorunlara neden olmaktadır. ÇHVK ilgili olarak ele alınması gereken bir başka konu da, hedefler arası ilişkileri öğrenme uygulamalarına dahil etmenin alternatif yollarını bulmaktır. Bu tez çalışmasında, yukarıda belirtilen konuları ele almak için denetimli öğrenme (DÖ), YDÖ ve özellik sıralama (ÖS) modelleri geliştirdim. Çok amaçlı en iyileme kavramlarından yararlanarak, işlenen değişkenlerin türüne karşı dayanıklı ve aynı zamanda hedefler arası ilişkilerden yararlanan öğrenme stratejileri önerdim. DÖ için, standart karar ağaçlarının çok amaçlı bir uzantısını ve seçici bir sınıflandırıcı zincir stratejisi önerdim. Önerilen modellerin kıyas modellerden daha iyi performans gösterdiğini deneylerle gösterdim. Ek olarak, çok amaçlı karar ağaçlarının, değişkenlerin etiket bilgisinin kısıtlı olduğu durumlarda daha iyi bir tahmin performansı göstermesini sağlayabilecek bir YDÖ modeli önerilmiştir. Yapılan deneylerle önerilen bu modelin kıyas modellerden daha iyi sonuç verdiği gösterilmiştir. Ayrıca, özelliklerde gözlenen yüksek boyutluluk ve ilgisizlik durumlarına karşın YD ağaçlara entegre edilmiş bir ÖS prosedürü geliştirilmiştir. Önerilen ÖS prosedürünün, kıyas yöntemlere göre tahmin performansını daha çok iyileştirdiği uygulamalarla gösterilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

c_i	Criterion for feature ranking score
c	Class type
C^j	Class set of target j
d_i	Distance from a node i to origin
d_r	Distance of node r to origin
D	Dataset
D_{test}	Test dataset
E	Loss function
G^s	Gini index for target s
G_r^j	Gini index value for target j in node r
i	Index of an instance
I_j	Information gain based on Shannon entropy
j	Index of a target
J	Size of an ensemble
h	A learning model
$h_t(x)$	Predictions of learning model h_t given training set x
L	Left
$l(y_t, h_t(x))$	Loss function of target t using learning model $h_t(x)$
M	Target cardinality
n_j	Number of instances in node j
N	Number of instances
p	Number of features in supervised datasets
p_{rc}	Relative class frequency of class c in node r
P	Feature cardinality
r	Index for node
R	Right
S_j	Instances in a parent node j
S_j^i	Instances of a child node i of parent node j

t	Number of targets
U	Number of unsupsevised criteria
x	Feature vector
x^n	Feature vector of instance n
X	Features
x_{ip}	Value of feature p of instance i
y	Target vector
$\widehat{y^n}$	Prediction vector of instance n for first step of training
$\widetilde{y^n}$	Prediction vector of instance n for second step of training
Y	Targets
Y_j	Column vector of target j
$\bar{Y}_j$	Average of target j
$\widehat{Y_m}$	Column-wise prediction of target j for first step of training
$\widetilde{Y_m}$	Column-wise prediction of target j
$\hat{y}$	Predicted value of target j
y_j	Value of target j
$\hat{y}_{ij}$	Mean response of target j in node i
y_{it}	Value of a target t of instance i
γ	Selected proportion of a parameter
γ_m	Proportion of features
γ_N	Proportion of instances
Λ	Covariance matrix
ξ	Feasible solution
$\hat{\xi}$	Efficient solution
Ξ	Feasible solution set
Ξ_E	Efficient solution set
$\hat{\upsilon}$	A nondominated point
Υ	Image of feasible set Ξ
Υ_{ND}	Nondominated set
Ω	Semi-supervised split score

Ω^s	Supervised scores for a split
Ω^u	Unsupervised scores for a split
$\in$	In
$\mathcal{O}$	Big O notation
$\|\Lambda(.)\|$	Determinant of Λ matrix of a node
$\|.\|$	Number of instances in a node
$\mathbb{R}^{N\times p}$	Real number with dimension N by p
$\mathbb{R}^{N\times t}$	Real number with dimension N by t

LIST OF ACRONYMS/ABBREVIATIONS

ANOVA	Analysis of Variance
CD	Critical Distance
CDMORF	Crowding Distance based Multi-objective Random Forest
DIS	Dismmilarity
ERC	Ensemble of Regressor Chains
ERF	Euclidean based Random Forest
FR	Feature Ranking
FS	Feature Selection
G	Gini index
MORF	Multi-objective Random Forest
MTD	Multi-target Dataset
MTL	Multi-task Learning
MTP	Multi-target Prediction
MTSC	Multi Target Stacking Corrected
RF	Random Forest
OOB	Out-of-Bag
PCT	Predictive Clustering Tree
PCTF	Predictive Clustering Tree Forest
PMORF	Pareto-based Multi-objective Random Forest
RMORF	Rank-based Multi-objective Random Forest
RRMSE	Relative Root Mean Squared Error
SC	Selective Chain
SIM	Similarity
SSE	Sum of Squared Errors
SSL	Semi-supervised Learning
ST	Single Task
TRF	Totally Randomized Forest

1. INTRODUCTION

Multi-target (or output) problems are wide-spread in different fields and present more challenging predictive tasks than single output cases (see Figure 1.1). For example, prediction of the tag categories for images are sometimes required in computer vision applications. In retail analytics, some forecasting problems require the sales estimations of multiple products. In predictive maintenance applications, prediction of a machine's failure time and its type might be required to prepare a maintenance plan. In all these examples, there is a need for prediction of multiple categorical values, numerical values and a mixture of both categorical and numerical values, respectively.

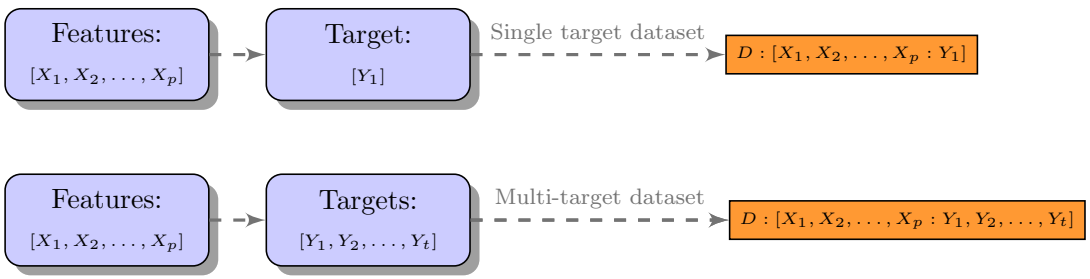

Figure 1.1. Single target and multi-target dataset illustrations.

In dealing with multi-target predictive tasks, some certain characteristics of those tasks need to be considered carefully. Initially, scale inconsistencies in the targets is a common property for multi-target datasets. Therefore, unless an independent and separate learning approach is employed for each of the predictive tasks, data transformation techniques are required in derivation of the learning models. However, this approach is problematic, as transformation techniques may change the statistical properties of the data. Besides, limited label information can be given as another property of multi-target datasets. In performing learning tasks for limited label settings, semi-supervised techniques are used. Majority of the semi-supervised techniques rely on distance-based calculations. However, scale inconsistencies and high-dimesionality in the features result in loss of effectiveness in distance-based calculations.

In addition to curse of dimensionality problem, another important challenge stems from irrelevancy and redundancy in features. Considering the label availability and scale inconsistencies issues, identification of the most useful features becomes more challenging yet more promising towards improving the predictive performance of the learning model.

Multi-target (multitask or multi-output) learning deals with simultaneous prediction of several outputs. In doing that, two main approaches are employed. Considering each target as separate learning problems is a trivial approach and known as local approaches [4]. That approach presents the most straightforward way of solutions as it treats the multi-target datasets as independent and separate learning problems [5, 6]. Since each target is learnt in isolation, those methods cannot make use of the potential of target interrelations. In order to address that issue, idea of using targets as additional inputs via stacking or constructing classifier chains to the feature space has recently become popular [5–8]. In those approaches, targets are learnt via separate learners and predictions of those learning models are added to the feature space, so that predictions of the targets take place as a part of feature space. Although target interrelations could take place in model building, still there is room for further improvement as those models do not consider the direction of the relations. To clarify, in stacking procedure all learnt targets are added as a whole (see Figure 1.2) or classifier chains are constructed with ensembles of random orders (see Figure 1.3), hence possible irrelevance of targets issue is completely ignored.

Global approaches can be considered as extensions of single target classifiers to their multi-target version [9–11]. Usually, this is realized by modifying the learning criterion to joint one. To clarify, for a single target problem, error score consists of values associated with that target. However, in multi-target case creating a joint learning criterion in a weighted sum form transforms the learning model to a global one. Despite those approaches inherently take into account the target relations, a considerable shortcoming of that category is that they require scaling or transformation operations when the targets are inconsistent in terms of their scales.

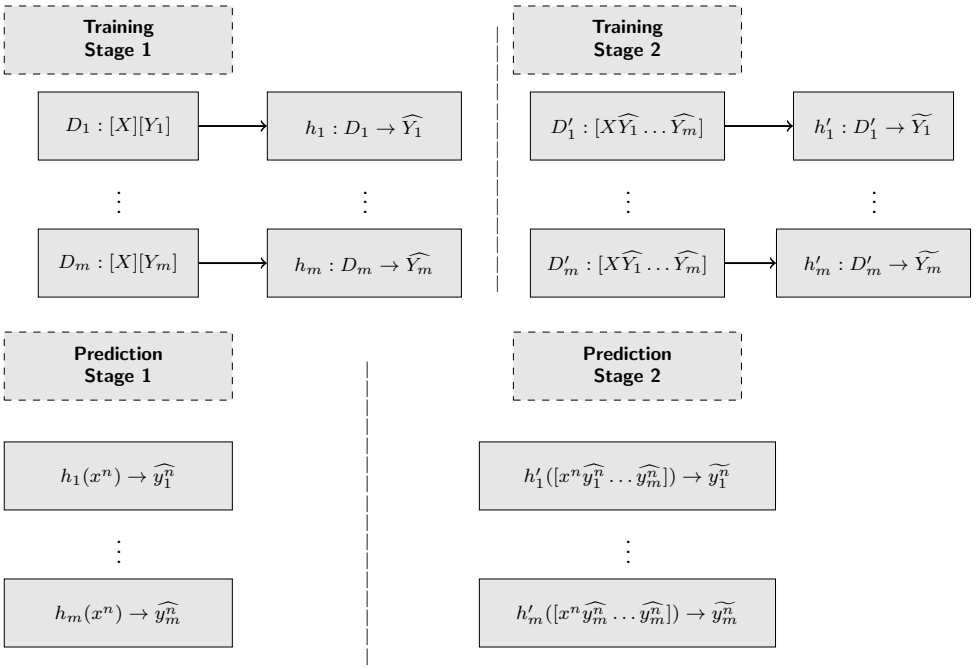

Figure 1.2. Stacking flow diagram.

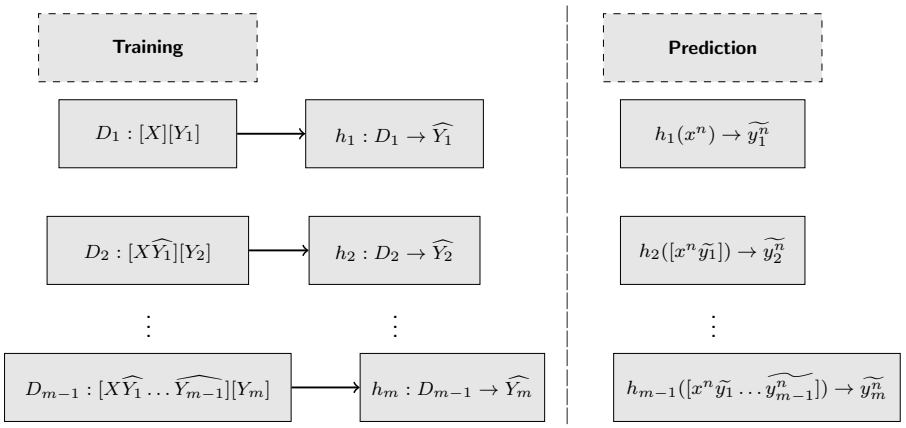

Figure 1.3. Chaining flow diagram.

However, scaling operations are criticized as they distort or change the distributional properties of the data [12].

Besides their structural properties, availability of label information is another issue with multi-target learning problems which makes predictive tasks more complicated [13–15]. To exemplify, considering computer vision tasks such as image tagging, an abundant amount of unlabelled (or anonymous) images can be easily found, however required labelled instance information is usually very limited compared to its unlabelled counterpart. Since supervised learners usually fail to deliver solid results when the target information is not enough, an integration of unsupervised and supervised approaches is utilized to handle predictive tasks for such environments [13]. That category is known as semi-supervised learning and usually shows superior performance compared to their supervised counterpart. However there are two major issues with attempts on addressing semi-supervised learning problems. First of all, when in semi-supervised setting global approaches are employed, this strategy would make learning model vulnerable to inconsistencies in targets [16, 17]. Second, semi-supervised models require processing of unsupervised information and in general those are distance or density based approaches to encode this information [18–20]. In doing that mostly on distance based calculations are preferred as they do not require any distributional assumption about the data. As a consequence, inconsistencies in features' scales challenge to define a distance metric that works well for such cases.

Another issue with features that deserve attention towards a better predictive performance is about refining the features, so that irrelevancy or redundancy are removed from features [21]. Those characteristics are undesirable as some classifiers, e.g. linear models, are vulnerable to correlated features. Besides, some classifiers, e.g. support vector machines, lose their effectiveness when irrelevant features exist in the dataset. Another challenging property stems from features is their high-dimensionality. In addition computational throughput, at higher dimensions position based properties, e.g. distances between instances, may become indistinguishable. Apart from that, reducing the number of features may increase interpretability and provide an easier understanding of a learning model, specifically for the black box ones.

By using features selection or ranking them, aforementioned issues could be resolved at the same time. Feature selection or feature ranking techniques from supervised and unsupervised perspectives are well-established in the literature [21,22]. However, contrary to those two perspectives, semi-supervised multi-target feature ranking and selection tasks are not explored in depth [23]. In capturing feature characteristics with a semi-supervised approach, aforementioned issues could be also resolved.

This thesis work aims to address the given problems for multi-target learning problems as separate three works. The scope of the works are as follows:

1.1. The Benefits of Target Relations: A Comparison of Multitask Extensions and Classifier Chains

The main theme of the first work covered in the thesis is about exploiting target relations without being disturbed by scale inconsistencies in the targets. The traditional approaches handle this problem by optimizing a joint score function over multiple targets such as weighted sum of the errors of each target. However this is problematic when the scale of the targets do not agree. To overcome the scaling issues, transformation techniques are widely employed in several studies yet they can change statistical properties of the data [12]. Considering the potential pitfalls of these operations, many applications make use of the modification of decision trees with multi-criteria (or multi-objective) learning strategies to learn a global model [4]. Although this revised design allows trees to enjoy target relations with a reduced computational cost, their performance is degraded when the targets are independent or irrelevant. To exploit the target dependencies as a source of side information, local models for each target are adapted by chaining strategies to expand the input space with the target information gradually.

Basically, chaining approaches train individual models to each target in some order and use the information from each model (i.e. estimations) to learn target interactions similar to a global model.

However, the success of the chaining approaches heavily depend on the training order and the estimation quality of the individual models. Chapter 3 introduces alternative tree-based learning strategies to handle the problem of target scaling and the order of the local learning in chaining strategies.

In the first proposal, the problems with target scaling are resolved using alternative splitting strategies which consider each objective in a multi-objective optimization framework. Here, we propose various strategies on deciding the split selection by evaluating the most promising trade-offs. Those promising solutions are referred as Pareto frontier and in multi-criteria domain, evaluations are usually based on that set of solutions. The second proposal deals with the problem of ordering in the chaining strategies. We introduce an alternative estimation strategy, minimum error chain policy, that gradually expands the input space using the estimations that approximate to true characteristics of outputs, namely out-of-bag estimations in tree-based ensemble framework. Our experiments on benchmark datasets illustrate the success of the proposed multitask extension of trees compared to the decision trees with de facto design especially for datasets with large number of targets. In line with that, minimum error chain policy improves the performance of the state-of-the-art chaining policies.

1.2. Semi-supervised Extensions of Multitask Tree Ensembles

In Chapter 4, we aim to handle multi-task designs for multi-target semi-supervised learning (SSL) problems that are robust to scale differences both in feature and target spaces. Similar to supervised cases, in order to handle scale inconsistencies data transformation is often employed, however in addition to possible computational burden, those operations may disturb the statistical properties of the data. SSL models require processing of unsupervised information where distance measures are widely employed. Use of classical distance metrics can be criticized as they lose efficiency when features exhibit type or scale differences. Besides, in higher dimensions distance metrics cause problems due to loss of discriminative power. Another problem with higher dimension feature spaces stems from difficulty with detecting and deciphering cluster forms. We propose a scale-invariant proximity measure with the use of tree-based ensembles.

This strategy preserves the original representation of the data.

We introduce alternative semi-supervised tree-based strategies to tackle learning problems with aforementioned issues. We update classical tree derivation procedure to a multi-criteria form to resolve scale inconsistencies. We define proximity based clustering indicators and extend the supervised model with unsupervised criteria. Our experiments show that proposed method significantly outperforms its benchmark learning model, that is predictive clustering trees.

1.3. An Ensemble-based Semi-supervised Feature Ranking for Multi-target Problems

Chapter 5 focuses on semi-supervised feature ranking (FR) applications for multi-target problems (MTP). Here, this work is a subsequent of the second one where we use the learning model proposed in the second work. We treat the split score function as a vector rather than unified weighted sum form to make it suitable for considering each criterion regardless of their scales. We propose a semi-supervised FR scheme embedded to multi-objective trees that takes target and feature contributions into account, simultaneously. Proposed FR score is compared with the state-of-the-art multi-target FR strategies via statistical analyses. Experimental studies show that proposed score significantly improves the performance of a recent tree-based and competitive multi-target learning model, i.e. predictive clustering trees. In addition, proposed approach outperforms its benchmarks when the available labelled data increase.

1.4. Contributions

Contributions of these works can be summarized as:

- In order to deal with given issues of scalarization, we borrow a multi-objective notion, that is Pareto optimality. We consider the learning model as a multi-objective optimization problem that respects the target trade-offs individually.

- We propose a controlled extension strategy that is based on evaluating whether an addition activity carries valuable information in learning of the rest or not. By doing so, we aim to prioritize the most informative tasks in learning process so that we create a better than random configuration by using a more comprehensive and flexible dependency relation.
- We derive a SSL strategy that can process unsupervised and supervised information without being tackled by the type of features and targets, simultaneously. Specifically, we consider supervised information obtained via processing the targets by means of multi-objective trees proposed in Chapter 3. In processing unsupervised information, usually distance based calculations required. However, distance based calculations may fail to provide reliable information for mixed type of features. In addition, at higher dimensions, those calculations may lose their efficiency in discriminative power. Here, we propose use of totally randomized trees in handling distance based calculations required for unsupervised information obtained via processing the features.
- We propose a feature ranking score that is able to heuristically capture and identify the semi-supervised characteristics of the data. In doing that, we use the semi-supervised tree structures proposed in Chapter 4 and propose an embedded feature ranking score collected during derivation of the trees. Proposed score is inspired by the quality measures used in evaluation of multi-criteria decision making domain. Here, we aim to identify the contribution of a feature by considering its characteristics from both supervised and unsupervised perspectives with the aid of the proposed unified score.

1.5. Organization of the Thesis Work

The thesis work is organized as follows: In Chapter 2, we provide the necessary background. In Chapter 3, we explain the proposed supervised learning models. Similarly, Chapter 4 is devoted to semi-supervised learning models, and Chapter 5 provides details on semi-supervised feature ranking study. Finally, the conclusions and future directions are given in Chapter 6.

2. BACKGROUND

This chapter introduces the notation used throughout the study and provides the key concepts together with formulations of the proposed and other benchmark methods.

2.1. Notation

Assuming a dataset D with N samples, let $X \in \mathbb{R}^{N \times p}$ and $Y \in \mathbb{R}^{N \times t}$ be the input space with p features and output space with t targets, respectively. Each (x_i, y_i) instance of dataset D consists of input vector of $x = (x_{i1}, ..., x_{ip})$ and output vector $y = (y_{i1}, ..., y_{it})$, where $i \in \{1, ..., N\}$. Each object i in dataset D associated with a set of fully observed y values where there is no ambiguity in labels. The objective of an MTL model is to learn model $h : X \rightarrow Y$ that maps a y vector with t values to each input item x by using the training dataset D.

2.2. Multi-objective Optimization

Scalarized multi-objective optimization problem refers to sum of task wise (or target wise) errors as a functional form of learning criterion. In case the learning criterion is not aggregated then scalarized error function for multi-target prediction tasks becomes a vector with t number of objectives (See Equation (2.1)).

$$E = [l(y_1, h_1(x)), ..., l(y_t, h_t(x))] \tag{2.1}$$

Noting that, E is defined for supervised tasks where l are defined as loss function. To clarify, if a target is continuous then l can be defined as sum of squared errors given the predictions of learning model h. Similarly, for a categorical target loss function l can be Gini index [24]. For unsupervised learning tasks, since the learning tasks are not based on predictions, l can be defined as a criterion that reflects the quality of the clustering, i.e. margin between data groups in terms of features.

Generally, a solution that minimizes all the objectives does not exist due to the conflicting objectives [25]. As a consequence, rather than looking for a single and global optimal solution, multi-objective optimization problems are solved for *set of efficient solutions*.

Let Υ denotes the image of the feasible set Ξ under the objective function mapping where $\Upsilon = f(\Xi)$, for a multi-objective optimization problem defined as [26]:

$$\begin{aligned} \min\ & f(\Xi) = [f(\xi_1), \ldots, f(\xi_t)] \\ \text{subject to}\ & \xi \in \Xi \end{aligned} \tag{2.2}$$

Definition 1: A feasible solution $\hat{\xi} \in \Xi$ is called efficient or Pareto optimal, if there is no other $\xi \in \Xi$ such that $f(\xi) \leq f(\hat{\xi})$.

Definition 2: If $\hat{\xi}$ is efficient, $f(\hat{\xi})$ is called nondominated point. If $\xi_1, \xi_2 \in \Xi$ and $f(\xi_1) \leq f(\xi_2)$ we say ξ_1 dominates ξ_2 and $f(\xi_1)$ dominates $f(\xi_2)$. The set of all efficient solutions $\hat{\xi} \in \Xi$ is denoted Ξ_E and called the efficient set. The set of all nondominated points $\hat{v} = f(\hat{\xi}) \in \Upsilon$, where $\hat{\xi} \in \Xi_E$, is denoted Υ_{ND} and called the nondominated set.

Definition 3: The collection of the nondominated elements of a decision space is denoted as *Pareto front*. Since no other element of the Pareto frontier can be preferred over the other one, then the aim of optimizing a multi-objective problem can be defined as approximating the Pareto set within the setting provided above. Figure 2.1 provides an example of a multi-objective optimization problem with two objectives and its Pareto frontier [1]. Noting that, performance criterion A and B represent two minimization objectives in Figure 2.1.

2.3. Decision Trees

Tree derivation process is based on partitioning of the data in a recursive manner [24]. Split function partitions the input space into its subsets.

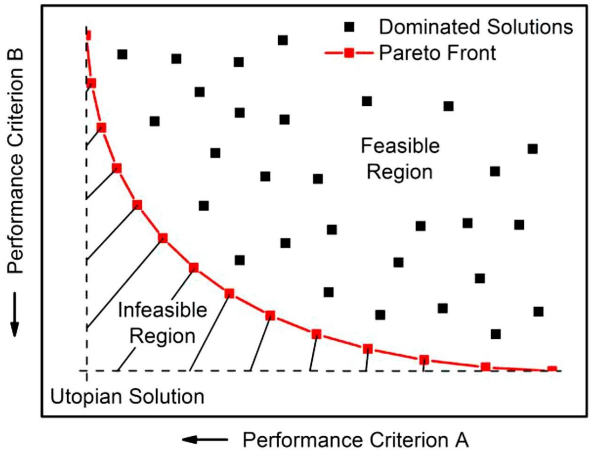

Figure 2.1. The example of the Pareto-optimal front [1]

The selected split results axis-aligned hyperplanes as it considers one variable at each time. The best partition value is selected considering a set of possible cut off values for the inputs and a quality measure associated with the candidate splits. Split quality is a function that is designed for providing information about the impurity reduction corresponds to the cut off value. Sum of squared deviations can be used as an impurity measure for a regression problem (see Equation (2.3)).

$$SSE_j = \sum_{i=1}^{N} (y_{ij} - \hat{y}_{ij})^2 \tag{2.3}$$

For a classification problem Gini index assesses the node impurities by using relative class frequencies, p_{rc}, $r \in \{left, right\}$ and $c \in C$ (see Equation (2.4)).

$$G = \sum_{c=1}^{C} (p_{rc}(1 - p_{rc})) \tag{2.4}$$

Following the split of a node, partitioning process repeats recursively for the child nodes until they meet a stopping condition (see Figure 2.3). These stopping conditions can be parametrized with maximum depth of the tree or minimum number of the elements appear in a node. Each leaf is associated with labels (see leaves at the bottom of the tree given in Figure 2.2). Labels refer to the prediction of a leaf node. Average of the leaf member associated with that target is assigned as the label of that target for a continuous output.

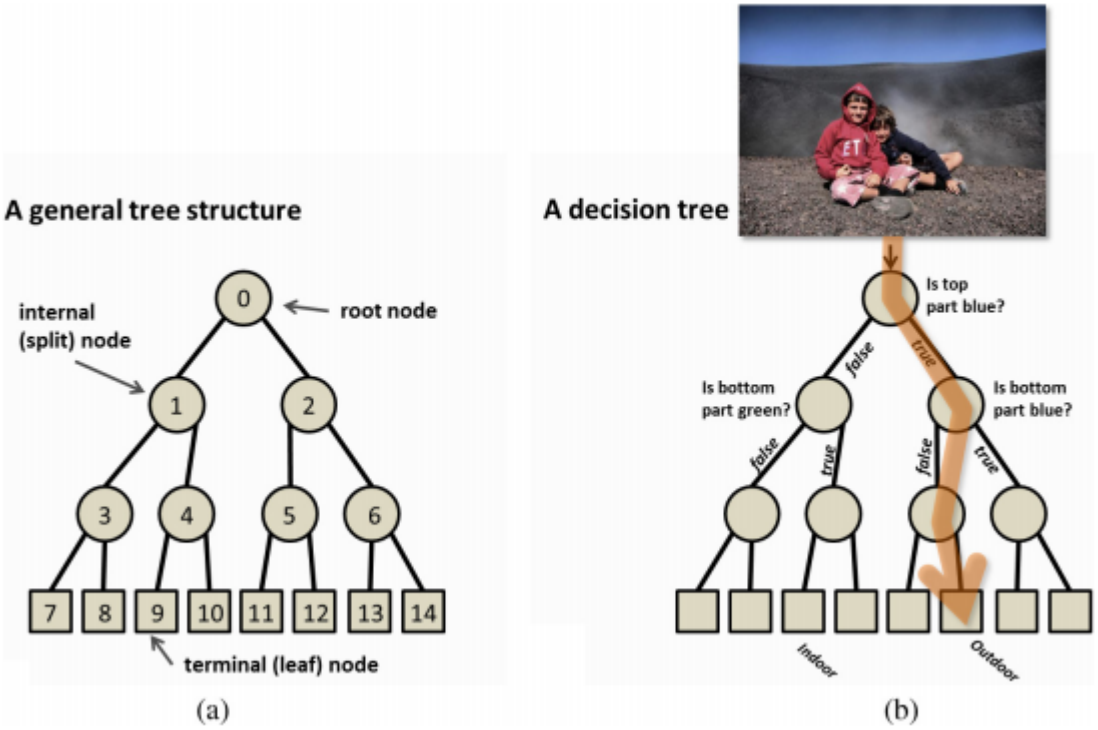

Figure 2.2. A general structure for decision trees and prediction by using them [2].

Likewise, the majority group of the leaf becomes the label for a categorical target. In order to make prediction for an unseen instance, it is enforced to traverse through the tree (see Figure 2.2 for prediction example). Label of the last visited leaf along the path yields test instance's prediction.

Require Training set $D = [XY]$ with n instances, stopping conditions
if termination criterion **then**
  Create leaf node and label it with its target averages;
  Return leaf node;
else
  Examine all possible binary splits for each of the input variables;
  Find the best split with respect to the quality measure;
  Apply the best split and create left node and right node;
  Return Recursive Partitioning(left node);
  Return Recursive Partitioning(right node);
end if

Figure 2.3. Recursive Partitioning Algorithm.

2.4. Random Forests

Decision trees are widely in use due to some certain advantages they have. Deriving a decision tree is a fast process and tuning its parameters is easy. Moreover, the flow of the split events and their interpretation is not hard for the modeler. In other words, by interpreting the structure of the decision tree, one can gain insights easily about the data. Additionally, extending the single output decision trees into a multiple output one depends on modification of the split function. To exemplify, a weighted combination of the impurity measure transforms the single response tree into its multiple response version. Nevertheless, decision trees are susceptible to have high prediction variance and that weakens the generalization power a single decision tree. In order to fix the shortcoming stems from this undesired property, one can utilize ensemble methods [27].

In order to improve the generalization power of the decision trees various methods are given in the literature.

Bootstrap aggregation (henceforth bagging) is one of these methods and the idea behind bagging is generating training datasets in a repetitively and randomly selective manner. Each bootstrapped data is associated with a classifier and for each classifier correspondent predictions are collected. For a regression problem averaging the collection values results the prediction. Likewise, for a classification problem majority vote is taken from the classifiers. Yet another classifier ensemble method is random forests (RF) that is a collection of independently derived decision trees. Using a random subset of the training data and considering a random subset of features at each split is the fundamental modification for a member tree of a random forest. Once the members are created, then the prediction task boils down to combining the results of each tree yields likewise in bagging.

Some certain properties of RF, e.g. out-of-bag (OOB) error, variable importance and margin maximizing property, make them preferable for predictive tasks. A useful concept of a random forest is, its OOB error estimations. Once a bootstrapped sample is drawn, a certain proportion of the data is classified as leftover or in other words these unselected data are the OOB instances of the bootstrapped sample. Overall prediction error can be computed by using all the data merely considering whether an instance is kind of OOB for a classifier or not. By using this OOB concept, the decision maker does not need cross validation any more that is a computational benefit of OOB estimates. Additionally, the data can be exploited to its full extent since each of the instance potentially contributes to both model construction and prediction phases.

Variable importance and proximity matrices are byproducts of a random forest setting and these concepts are the natural extents of OOB predictions of the trees. In order to come up with an evaluation of the variables in terms of their importance one can make use of OOB errors as follows: after calculating OOB error of the default setting, a feature is selected and only this selected feature is permuted. OOB error is calculated one more time and an importance value is assigned to the feature of the interest with regarding to the difference between these two OOB errors.

Proximity matrix refers to similarities between the instances and by using this matrix one can obtain a gauge that serves as distances between the instance pairs. In order to obtain the proximity matrix each instance is pushed through to the associated trees and whenever a pair of instance meets in the same terminal node of a decision tree then their proximity is increased by one. And finally off diagonals of total proximity matrix is divided by number of trees. This proximity matrix is also easily converted to a distance measure by subtracting the matrix components from 1 and taking square root of the results.

And finally, maximum margin behavior is another important property of random forests and it can be defined as follows: when a classifier is created for a random sample, the boundary it provides passes through a less dense region. Intuitively, the margin that boundary yields does not need to be optimal since any decision boundary passes within this gap results the same value for the entropy function. What makes the random forest having maximum margin property is, it uses a combination of the randomly created decision trees' wisdom to make conclusion about an unseen data [2].

2.5. Decision Trees for Unsupervised Learning Tasks

Unsupervised learning is utilized in various descriptive tasks such as clustering, density estimation and dimensionality reduction. Among several types of unsupervised learning tasks, our focus is on *clustering*. Discovering homogeneous groups constitutes main concern of clustering tasks [24]. Clustering algorithms require similarity or dissimilarity information between instances to assign similar ones to the same subgroup. Likewise in supervised learning, an indicator that measures how well clustering captures the data characteristics, is optimized. Here, various concerns and associated indicators can be given. To exemplify, instance assignments to the subgroups is a good practice when within cluster variation is reduced by doing so. To measure deviation, differences between cluster medoids are employed where medoid of a cluster represents the instance with lowest distance to the samples that are in the same group. As another example, well separation of those groups is preferred, hence distance between cluster medoids should be increased whenever possible.

In order to quantify within cluster deviation or separation information, different measures are used. In addition to classical distance based measures, similarity or dissimilarity based ones are widely employed in clustering. To clarify similarity, it refers to strength of the relations between samples [28]. Correlation and proximity are typical representatives of similarity. Among various alternatives, proximity values between instances provide a flexible measure when trees are used in their derivation. Due to the given property, proximity measure enjoys the benefits of decision trees, i.e. be able to capture nonlinear relations, can handle data with scale differences and robust to high dimensionality issues. In order to obtain in an unsupervised way, total random forests is an efficient alternative [29]. Unlike supervised decision trees, in a tree of a total random forest, a random split for a random feature is directly selected without being evaluated from a performance perspective. Once a total random forest is generated, for each pair of instances number of their meets at the same leaf is counted and then normalized by forest size. Following that, a dissimilarity information can also be derived as suggested in [27], and can be replaced with any distance based measure used in clustering applications.

Another idea of benefiting from trees in unsupervised learning appears in density estimation tasks [2]. Likewise in derivation of supervised trees, unsupervised trees try to find a good approximation to the underlying distributions. Following work of [2], at each node split unsupervised entropy is aimed to reduce. To achieve that, they propose a Shannon entropy based information gain indicator which is summarized in Equation (2.5):

$$I_j = log(|\Lambda(S_j)|) - \sum_{i \in \{R,L\}} \frac{|S_j^i|}{S_j} log(|\Lambda(S_j^i)|) \tag{2.5}$$

In Equation (2.5), Λ, $|\Lambda(.)|$, and $|.|$ refers to $d \times d$ covariance matrix, determinant and cardinality of nodes, respectively. L, R, indexed by i for S, stands for left and right child of the parent node S_j. Noting that, they assume that data can be modelled with a mixture of Gaussian distributions, hence the use of given structure is limited with continuous form feature sets.

Figure 2.4 represent a toy application of unsupervised trees for a given training dataset [2]. Figure 2.4 represents use of unsupervised decision trees for density estimation purpose. In derivation of those trees, it is assumed that the data can be modelled via a mixture of Gaussian distributions. To clarify, three trees with different depths are trained in Figure 2.4. Here, obtained leaf nodes are used to estimate parameters of Gaussian distributions in modelling the data.

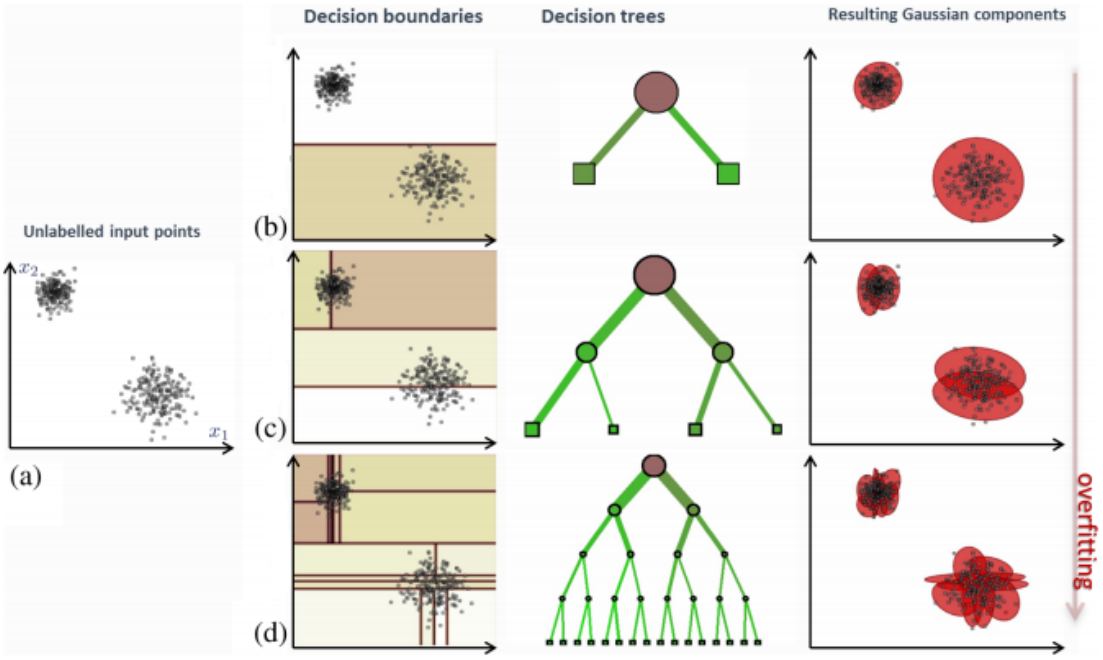

Figure 2.4. Unsupervised trees and associated partitions for various depths [2].

2.6. Semi-supervised Learning

Semi-supervised learning (SSL) is in the intersection of two learning schemes, namely supervised and unsupervised learning models. Target based learning is classified as supervised, whereas no target correspondence implies a need for unsupervised techniques. A third variant is the semi-supervised learning, where amount of labeled data is limited compared to unlabeled data on hand, and utilizes both of these sources in order to consolidate the model for unseen data [30]. Success of semi-supervised models generally depends on satisfying some certain conditions [13].

Considering a pair of instances, smoothness assumption requires the instances to have similar labels if they are close in features' space. Low density (or clustering) assumption guarantees that, the decision boundaries pass through less dense areas of the features' space. Manifold assumption is another condition that requires to be satisfied and refers to have a lower-dimensional representation for data compared to its original input space.

The two main branches on SSL taxonomy are based on inductive and transductive methods [13]. Transductive methods consist of graph based methods and generally similarity graphs of the data points are exploited in identifying the weighting schemes and inferences. Categorization of inductive methods is based three main groups. First group is wrapper methods and those methods are generally developed on the training datasets which are iteratively enlarged by the predictions of the classifiers. Self-training is a wrapper method where a single learning model is trained over an initial dataset. Once the dataset is enlarged with classifier's predictions, several rounds of trainings are realized by using updated training sets. Co-training is another category of wrapper methods which is similar to self-training. In co-training applications, usually datasets are divided into two groups and independent classifiers are derived by using those datasets. Upon completion of a training phase, the most confident predictions of each group are selected and added to the associated group.

Second category of inductive methods can be given as unsupervised preprocessing. In those methods, usually labelled and unlabelled data information are used in consecutive steps. To exemplify, majority of feature extraction methods can be given as applications of unsupervised preprocessing. Besides, cluster-then-label method is another strategy used as unsupervised preprocessing where resultant clusters are employed for augmenting the classification process. Third category of inductive methods consists of intrinsically semi-supervised methods. To exemplify, methods consider max-margin property such as support vector machines, naturally exploit low density (or clustering) assumption of SSL.

3. THE BENEFITS OF TARGET RELATIONS: A COMPARISON OF MULTITASK EXTENSIONS AND CLASSIFIER CHAINS

3.1. Introduction

During the last decade a vast range of techniques for multi-target prediction (MTP) have been proposed [4]. Taxonomy of MTP models is based on two categories, which are defined as *local models* and *global models*. In addition, *problem transformation* and *algorithm adaptation* are respectively substituted for these categories. In local approaches, separate learners for each of the target and a collection of outcomes from individual models are required. Unlike local approaches, global models are one-off learners, in other words values returned by a global model cover all necessary predictions (see Figure 3.1). None of these approaches can universally outperform the other, indeed, both have certain merits and limitations. Global models require adaptation of local learners to provide joint prediction of several targets. This adapted design permits global models to enjoy task relations during training phase therefore these models correspond to an important learning paradigm known as *multitask learning* (MTL). From taxonomy perspective, MTL is a particular group of transfer learning and benefits from transferring information between tasks [31]. In MTL, task relatedness is exploited in various forms. To exemplify, task relations can be based on either the assumption that they share a common representation [32] or the assumption that there is a relation between their parameters [33]. Moreover, from side information perspective, these assumptions can be considered as prior forms where local models lack [34].

Local models frequently have higher computational costs than global models. As the number of targets grows, it gets correspondingly more complicated to train local models. In addition, global approaches produce more compact models compared to the set of individual learners.

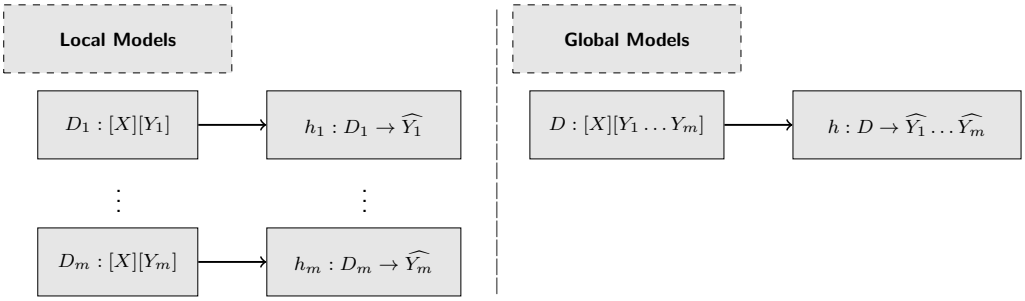

Figure 3.1. Local and global modelling in multi-target prediction framework.

From generalization point of view, global models tend to present better results since they can benefit from statistical dependencies between targets [9]. Noting that, exploiting target relations is not limited with global models. Local approaches can also utilise target dependencies yet they require tailored strategies like *stacking* and *chaining*.

Despite the given advantages of global approaches, local models outperform the others when a dataset's outputs are irrelevant or independent. Enforcing dissimilar learning tasks to share the same learning structure results degradation in predictive performance of global classifiers [33]. This phenomenon is known as *negative transfer* and badly affects the predictive power of the learner [31]. In addition, global models become more problematic when the targets exhibit scale differences. To address the predictive tasks with scale differences, various data transformation techniques are employed. However, transformation operations may ruin the statistical properties of the data [12].

An independent learner for each target is the standard local strategy for MTP tasks. This policy is immune to scale differences and can only exploit input-target dependencies. Use of the targets or their estimations as additional inputs is a simple approach to integrate target-target relations with the training process. This approach implicitly introduces the conditional label dependencies from a probabilistic point of view as discussed by [**?**].

It is argued that unconditional and conditional label dependencies are different than each other and modelling these dependencies lifts the predictive performance of the learners [**?**]. They show that local meta-learners can model these dependencies, namely *stacking* can be considered as a modelling instance for the former. Likewise, *classifier chains* are representatives for the latter type of dependency. From a deep-learning perspective, local meta-learners serve as nodes of inner layers and hence the proposed design allow them to behave alike deep learners [**?**, **?**]. And lastly, when target-target relations are discovered, they deliver a form of knowledge which can be used in reinforcing the predictive performance from side information perspective [4].

The main focus of this study is to develop mechanisms to capture and exploit target relations in MTP models that are robust to scale differences in targets. We employ random forests as they construct competitive benchmarks for both local and global strategies. Specifically, we used predictive clustering trees [9] and ensembles of random chains [5] as benchmarks and measured the impact of applying our methods to MTP tasks. Predictive clustering trees consider a unified score measure in split evaluation during tree growth. We replace this score measure with a multi-criteria one which allows us to keep the targets and therefore corresponding scores in their original scales. To clarify, multi-criteria score handle targets with different scales including mixture of target types. By doing so, we aim to preserve statistical properties of the data throughout tree derivation. Moreover, since our model is an instance of multitask learning model therefore target interactions inherently take place in model.

Ensembles of regressor chains proposed in [5] require a limited number of random sequence in chain construction. Another chaining strategy given in [6]'s study is an alternative yet they only consider linear dependence among targets. Classifier chains need individual models for targets, hence scaling is not a matter of concern. Target relations are exploited by introducing their estimations as additional inputs. We propose a flexible and entirely data-driven chaining strategy for chain order derivation. It is flexible since it hints about a more general dependency than correlation and can identify relations targets including mixtures of descriptive and numerical values.

This strategy is not entirely a multitask learning model yet it still utilizes target relations in a one-way manner as targets are treated as additional features.

This study proposes a multiobjective learning framework for MTP and an alternative input space expansion method that benefits from the task relatedness. Our experimental results show that proposed approaches either significantly improve or perform at least as good as their benchmarks in terms of predictive performance. Our results encourage further exploitation of task relatedness in learning processes. In addition, our findings support the need of data-driven information transfer structures for better learning practices.

The rest of the chapter is organized as follows; Section 3.2 provides related work and Section 3.4 delivers the algorithms proposed. Section 3.5 introduces the experiments and discusses the results.

3.2. Related Work

The survey by [4] summarizes the works on MTP problems and offers several different settings as a learning problem. The methods are basically described under five categories. They consider *i)* independent models as the standard method for MTP and learn each target in isolation. They are local models that cannot exploit target relations. *ii)* Similarity enforcing methods are based on the communality assumption of task representations or parameters. *iii)* Relation-exploiting methods require side information like hierarchies, dependencies or graph representations. *iv)* Relation-constructing methods discover the relation by themselves, this information is not given unlike previous groups. In addition, kernel methods consist *v)* representation-exploiting models.

Each of these method groups have certain benefits and drawbacks. Independent models miss target relations however they show superior performance when the targets are irrelevant. In similarity enforcing models, a joint learning criterion in a weighted sum form is usually defined and optimized.

They enjoy power of well-established optimization techniques yet require data transformation when target scales are different. Relation-exploiting methods benefit a priori target interrelations hence cannot be applied for problems with no priors. Relation-constructing methods allow various forms of relations such as task clusters, hierarchies or classifier chains based on training data. And finally kernel methods are criticized due to computational costs.

Our proposals benefit from the similarity enforcing and relation-construction methods. Therefore, this section focuses on the comparison of the existing literature with the proposed methods. This comparison is restricted to the decision-tree based multitask learning approaches as our proposals utilize decision trees as base learners.

Learning decision trees in multitask setting appears in a number of papers [10, 11, 35–37]. These studies optimize a joint learning criterion greedily throughout the decision tree learning. In other words, splitting is performed using a weighted sum of scores of each task in forest setting by [10, 11, 35, 36]. Multi-objective ensembles are reported to perform better than single objective counterparts in these studies. As an alternative strategy, [37] constructs population of randomized trees and they introduce an evolutionary algorithm to refine the ensemble using the Pareto optimal regressors. Node cardinality is considered as an additional score in their study for generalization purposes (i.e. to avoid overfitting).

Despite the simplicity in implementation, these multitask decision trees have certain issues to handle. An aggregated learning criterion needs hyperparameter calibration in advance. In addition, greedily optimizing the aggregated function may result overfit in tree setting [37] and reduce diversity of the trees. Moreover, merely optimizing the selected weight configuration may result missing to explore potential trade-offs between tasks. Scaling may resolve some of those problems however it comes with a price of errors in measurements and inference derivations [38, 39].

Exploitation of task relations is considered in many MTL approaches.

Besides MTL, the target values used as predictor is an example of side information which is based on task relatedness [5]. In this approach, task interactions do not need to be symmetrical contrary to the MTL. For example, *stacking* is a procedure that benefits the task relations. It initially constructs independent classifiers for each of the targets by using the inputs only. Following step 1, new learners are separately derived by using the target-extended inputs for each task (see Fig 1.2). Stacking diverges from MTL since all the information gathered from outputs is stored in the input space where the knowledge transfer is allowed from the new and transformed input space to a single task. Gradually *chaining* the targets in a random [5], a lattice [7] or a predetermined order [6] is another way of utilizing the task relatedness (see Figure 1.3). Limited number of target orders to construct an ensemble is considered in [5]. The correlation chains are proposed by [6]. Both of these studies are instances of local models and designed for regression tasks. Methods given in [5] can be criticized as they randomly generate the target orders and they may cover the best target order for small number of targets case. Study given in [6] is an attempt to obtain chains better than random orders by considering the target correlations. They only consider the correlations between targets and employ an add-and-learn chaining strategy. However, use of a linear measure to quantify the correlation between targets is problematic when the target relations are not linear. More importantly, they add the actual target values in training phase although the real target values are not available in real prediction scenario. As a consequence, learning models resemble the training dataset rather than a set that is more similar to unseen examples.

3.3. Methods

We refer Chapter 2 and information given therein for key concepts used in proposed methods and benchmark studies. Assuming a dataset $D = \{(x_1, y_1), ..., (x_N, y_N)\}$ with N by $p + t$ dimensions, single task (ST) or independent type of learning refers to separate derivation of t number of h_j models approximating the true output Y_j best. In order to extend the ST model to an instance of MTL model, we employ multi-objective adaptation of the selected base learner.

We implement the following two models, i.e. multi-objective random forest, and selectively chained random forest.

3.3.1. Multi-objective Random Forest

In order to promote the diversity and handle the targets with different scales in a natural way, we employ Pareto-optimality approach in split selection step of the tree derivation procedure. Once the candidate splits are created, objective vector regarding to sum of squared errors and Gini index are calculated for numerical and categorical targets, respectively. In order to extract the solutions that approximate the Pareto frontier, we employ nondominated sorting procedure [40]. Nondominated sorting helps to segment the obtained solution set into layers (frontiers). This procedure creates the frontiers regarding to dominance relation between the solutions. According to the non-dominated sorting procedure given in [40], each solution is compared with others in terms of domination and the number of times a solution is dominated are recorded. If no other solution dominates a specific solution, then it becomes the member of first layer. In that sense, first frontier approximates the Pareto front the best. Figure 3.2 illustrates the results of the non-dominated sorting procedure for an optimization model with two minimization objectives.

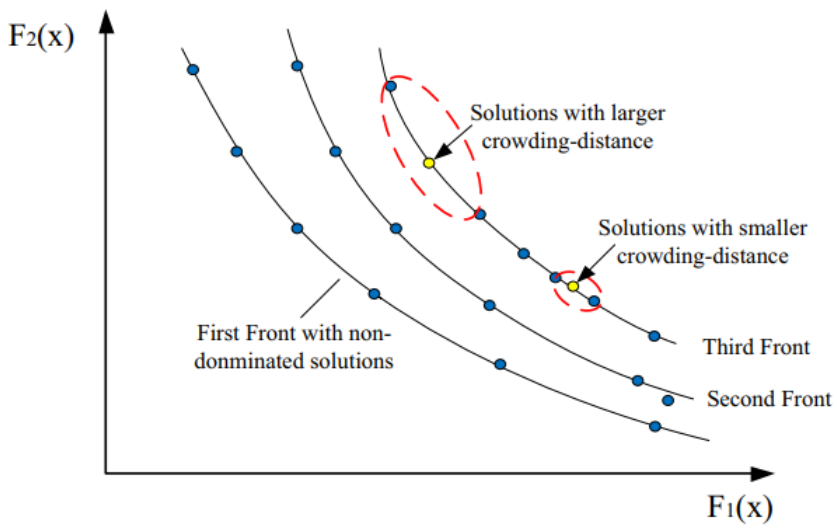

Figure 3.2. Ordering of the solutions with non-dominated sorting [3].

Once we obtain the first frontier we employ three split selection strategies. In the first strategy, we pick a random split from first frontier (see Figure 3.3). Corresponding classifier ensemble will be referred as Pareto based Multi-objective Random Forest (PMORF). As a second alternative we compare the crowding distance of the solutions and pick the solution with maximum crowding distance. The associated learner group will be referred as Crowding Distance based Multi-objective RF (CDMORF). Crowding distance is a metric we borrow from multi-objective evolutionary algorithms and basically it aids to maintain the diversity as it favors the solutions from less dense areas [40]. In doing that, Euclidean distances between neighbor solutions are used for an m-dimensional space (see Figure 3.3).

Require Solution set Q
$q = |Q|$;
for Each member of the solution set **do**
  Set $Q[i]_{distance} = 0$;
end for
for Each objective m **do**
  $Q = \text{sort}(Q, m)$;
  Set distance of the first and the last solutions to ∞;
  for $i = 2$ to $q - 1$ **do**
    $Q[i]_{distance} = Q[i]_{distance} + (Q[i+1].m - Q[i-1].m)/(f_m^{max} - f_m^{min})$;
  end for
end for

Figure 3.3. Crowding Distance Calculation Algorithm.

And the last selection alternative is based on comparing overall the ranks of the solutions and henceforth it will be Rank-based Multi-objective RF (RMORF) (see Figure 3.4). For each target we assign ranks to each split candidate considering the quality measure they produce.

By doing so, each split candidate has a *position label*. Once we obtain the position information of a split candidate, we pick the solution with lowest average in terms of rank values.

Each of these evaluation methods transforms the ordinary tree derivation process into an instance of multitask learning hence utilize the task interrelations. In addition to that, they are insensitive to scaling differences of the targets and do not need special handling for such cases. Figure 3.5 is fed with multi-objective trees, hence it results in a multi-objective random forest. Noting that, the aforementioned split selection policies create different types of multi-objective random forests.

Require Training set $D = [XY]$ with n instances, stopping conditions
if termination criterion **then**
  Create leaf node and label it with its target averages;
  Return leaf node;
else
  Draw a S^* random subset of the input variables;
  Examine all possible binary splits for each of the input subset S^*;
  Create the set of first frontier with respect to the quality measure in vector form;
  Select a random split from first frontier;
  Apply the best split and create left node and right node;
  Return Recursive Partitioning(left node);
  Return Recursive Partitioning(right node);
end if

Figure 3.4. Base Multi-objective Tree Algorithm.

Require Training set $D = [XY]$ with n instances, stopping conditions, ensemble size

forest=$\emptyset$;

for Ensemble size **do**

Create a D^* random sample of the training set D;

regression tree $\leftarrow$ Base Multi-objective Tree Algorithm(D^*);

forest $\leftarrow$forest $\cup$ regression tree;

end for

Figure 3.5. Base Multi-objective Forest Algorithm.

Require Training set $D = [XY]$ with n instances, stopping conditions

if termination criterion **then**

Create leaf node and label it with its target averages;

Return leaf node;

else

Draw a S^* random subset of the input variables;

Examine all possible binary splits for each of the input subset S^*;

Create the set of first frontier with respect to the quality measure in vector form;

Select the split with minimum average rank;

Apply the best split and create left node and right node;

Return Recursive Partitioning(left node);

Return Recursive Partitioning(right node);

end if

Figure 3.6. Rank Based Tree Algorithm.

3.3.2. Selective Chained Random Forest

Using the targets as additional inputs change the trajectory of the learning for the better yet its implementation needs to be handled carefully. In order to cope with the shortcomings of the state-of-the-art models given in Section 3.2, we prefer to create the chains in a more principled way rather than randomly generating them. It is desired to avoid an exhaustive search for the best chain configuration, hence a simple and data driven selection strategy is an obvious need specifically for larger number of targets. Correlation matrix is a natural option yet its drawbacks are given above.

In line with this purpose, we propose a selective chaining heuristic. In this strategy, the target orders are based on the observed OOB errors of the candidate target sets. Target with the lowest OOB error is selected and its OOB predictions are appended to the input space. We refer to Section 2.4 for details in calculating OOB errors for a given ensemble. It proceeds recursively until each of the target has its own learning model. See Figure 3.7 for an illustration of selective chaining algorithm (Figure 3.8). Ensemble learning models are promoted to used in such space expansion strategies as they aid to reduce the additional noise to the ingredients [5]. In this study random forests are used as the base learners. In addition to this, ensemble models allow to create and use the predictions, namely OOB predictions, that resemble unseen instances in training phase. Adding the ground truth values of the targets is an option yet in testing phase targets are not available to the model (see Figure 3.9). Keeping in mind that all the targets other than the first added one, needs the information of its predecessors. As a consequence, using the models that are akin to the unseen instances hints for a better practice.

An advantage of selective chaining heuristic appears in the best target configuration search for datasets with a large number of targets. Another advantage of such strategy can be given from outlier task point of view. A controlled expansion strategy that pushes the most irrelevant target would be more beneficial and the proposed selection algorithm intuitively operates in that way.

Figure 3.7 represents the way selective chaining algorithm trains dataset D with three outputs. Initially, three separate models are built for each target and the target with the lowest OOB error is selected. In the example above second is picked and its OOB predictions are concatenated to X space. By using extended input space independent models are derived for the leftover targets, namely first and third targets. The same selection procedure holds and third is selected next. OOB predictions of third target are appended to the current space. The last model is created for first target and process ends.

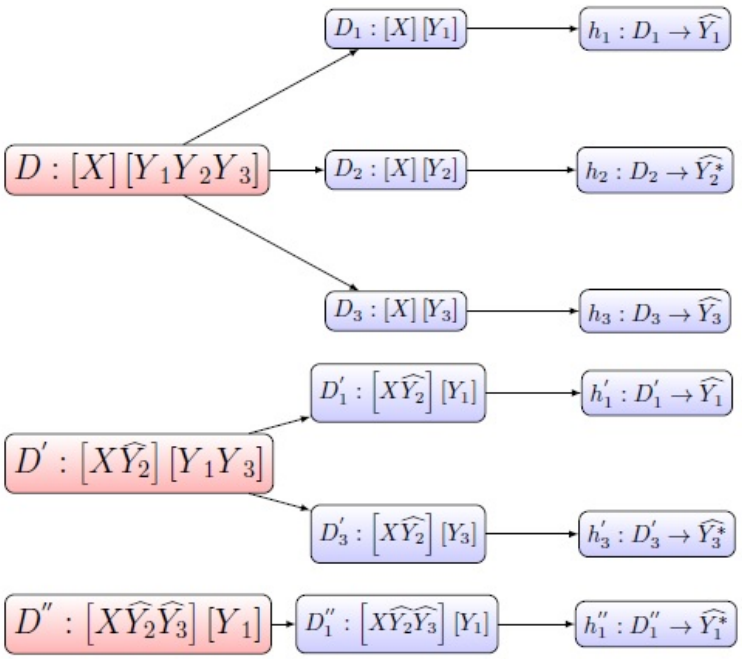

Figure 3.7. Selective chaining flow diagram.

Require Training set $D = [XY]$ with n instances

$S = \{1, \ldots, m\}$;

while $S \neq \emptyset$ **do**

  for $j \in S$ **do**

    $h_j : D \rightarrow Y_j$;

    Calculate OOB error($h_j(X)$);

  end for

  $k \leftarrow$ argmin(OOB error($h_j(X)$);

  $\hat{Y_k}$ = OOB prediction($h_k(X)$);

  $X \longleftarrow X \cup \hat{Y_k}$;

  $D \longleftarrow X \cup Y$;

  $S = S \setminus k$;

end while

Figure 3.8. Selective Chaining Algorithm.

Require Test instance $\tilde{x}$, chain models h_j, $j = 1, ..., m$

$\tilde{y}$=$\emptyset$;

for $j = \{1, ..., m\}$ **do**

  $\tilde{y_j} = h_j(\tilde{x})$;

  if $j < m$ **then**

    $\tilde{x} \longleftarrow \tilde{x} \cup \tilde{y_j}$;

  end if

end for

Figure 3.9. Selective Chaining Prediction Algorithm.

3.3.3. Algorithmic Complexity

Computational complexities of a multi-objective and an ordinary decision tree differ with respect to the split selection policy used. Complexity of a Pareto based multi-objective tree is $\mathcal{O}(MP^2N\log N)$, where M is the number of targets compared at each split, P is the number of features and N is the number of instances considered. Algorithmic complexity of a Pareto based Multi-objective Random Forest (PMORF) with size J becomes $\mathcal{O}(JMPN\log N)$ as an RF requires $\sqrt{P}$ number of features for evaluation purpose. Crowding Distance based Multi-objective RF (CDMORF) involves additional calculations of the distance metric hence its overall complexity is $\mathcal{O}(JMN\log N(P+\sqrt{P}\log\sqrt{P}))$. Likewise, Rank based Multi-objective RF (RMORF) requires a sorting procedure for each of the targets, hence associated complexity is $\mathcal{O}(JN\log N(MP+\sqrt{P}\log\sqrt{P}))$.

Considering the Selective Chaining (SC) procedure, we perform $\frac{M(M+1)}{2}$ many times single target RF, hence overall chain estimate results a complexity of $\mathcal{O}(J\frac{M(M+1)}{2}$ $(P+\frac{M}{2})N\log N)$. Following [5], $\frac{M-1}{2}$ number of additional targets take place as input in a chained model on the average. Prediction time takes M many times traversing J number of trees in the forest for SC. For the multi-objective RF variants, prediction complexity is defined J many times tree traverse.

3.4. Experiments and Results

3.4.1. Experiments

All 18 datasets used in these experiments are publicly available and they are accessible through the websites given in [6]. Dataset identifications are given in Table 3.1. In order to compare our proposed algorithms we used the methods presented in [5] as their implementation framework is open and moreover, their work provides competitive performance benchmarks. The base learners in [5] are also bagging trees which is similar to the proposed ensembles.

In line with that, we select Ensemble of Regressor Chains (ERC) [5], Single Target (ST) [5], Multi Target Stacking Corrected (MTSC) [5] and Multi-objective Random Forests (MORF) [36] for our experiments. We refer the reader for the details of the algorithms. These methods are implemented by [5] in WEKA and we use the software provided by the authors in [5]. We employ the same parametrization given in [5] for these algorithms, i.e. bagging of 100 trees with 10-fold cross validation. For ERC 10 chains are generated for the ensemble. And finally, size of the ensemble set to 100 for MORF method. We propose four methods: Pareto based Multi-objective RF (PMORF), Crowding Distance based Multi-objective RF (CDMORF), Rank based Multi-objective RF (RMORF), and Selective Chaining (SC). We also set the ensemble size to 100 for our methods. We perform ten-fold cross validation with five repetitions. Our proposals are implemented in MATLAB R2014A and the codes are shared in a github project for reproducibility purposes [41]. In addition to the codes, we share the training datasets we used throughout the experimental study. For the sake a fair comparison, we used the same training and test sets regarding to methods without any exception. And finally, no data preprocessing performed.

In line with [5], Relative Root Mean Squared Error (RRMSE) is used as the performance indicator of the algorithms. RRMSE of a given model h_j is computed as:

$$RRMSE(h_j, D_{test}) = \sqrt{\frac{\sum\limits_{(x,y)\in D_{test}} \left(\hat{y}_j - y_j\right)^2}{\sum\limits_{(x,y)\in D_{test}} \left(\bar{Y}_j - y_j\right)^2}} \tag{3.1}$$

In Equation (3.1), $\hat{y}_j$ is the predicted value of model h_j for target j. In addition to that, y_j refers to the ground truth of the test set's target j and $\bar{Y}_j$ is the average of training dataset's target j.

Table 3.1. Datasets.

Dataset	**Samples** (N)	**Features** (p)	**Targets** (m)
EDM	145	16	2
ENB	763	8	2
Jura	359	15	3
Slump	103	7	3
Water quality	1060	16	14
OES10	403	298	16
OES97	334	263	16
ATP1D	201	411	6
ATP7D	188	411	6
Andro	49	30	6
Wisconsin Cancer	198	34	2
Stock	950	10	3
CalHouse	1032	7	2
Puma8NH	2457	6	3
Polymer	41	10	4
M5SPEC	700	80	3
MP5SPEC	700	80	4
MP6SPEC	700	80	4

3.4.2. Results and Discussion

To find out whether a statistical significance of the difference between methods exists or not, we employ the methodology given in [42]. Our experiments consist of multiple algorithms with multiple datasets. For such experiment settings Friedman test, a nonparametric version of ANOVA, is recommended in [42]. Friedman test uses the average ranks of the algorithms. We consider two settings for the comparison of predictive performances. In per dataset analysis we consider the errors that belong to a dataset and use the average of them for comparison purpose. In per target analysis, we pool the errors regarding to targets and compare targetwise results separately. In order to find out which methods are significantly different from the others, we perform Nemenyi test. In this test, all the methods are compared with each other in terms of their average rank. If the difference between a pair of classifiers is at least different than a critical value, then these two performances are significantly different than each other. Running Nemenyi test produces a difference matrix and by using it a compact representation of the results is given on a scale with groupings. To clarify, methods that are not significantly different than each other are tied with line. All the results can be seen in Table A.1 and its subsequent tables given in Appendix A.

3.4.2.1. Screening of Multi-objective Alternatives. We tested the multi objective methods to find out the one with the best performance amongst alternatives. Figure 3.10 reveals that, there is no significant difference in multi objective extensions, yet, RMORF has the lowest average rank. In other words, RMORF shows superior performance than its competitors in terms of predictive performance. Hence we proceed our experiments by considering RMORF. In addition we include MORF in our experiments as it is state of the art benchmark of multi objective methods.

It is worth mentioning that, any of the multi-objective strategy outperforms MORF. Results in Figure 3.10 support alternative tree modelling approaches in split evaluation as they enhance predictive performance of classical tree splits with weighted sum of errors.

This result encourages to implement various ranking and selection in split evaluation during tree growth.

3.4.2.2. Per Target Analysis. Considering per target performances, there is a statistically significant evidence to reject equivalence of the algorithm performances ($p = 1.106e-6$) (see Figure 3.11). Considering the results for Nemenyi tests, we obtain a critical difference (CD) value equal to 0.76.

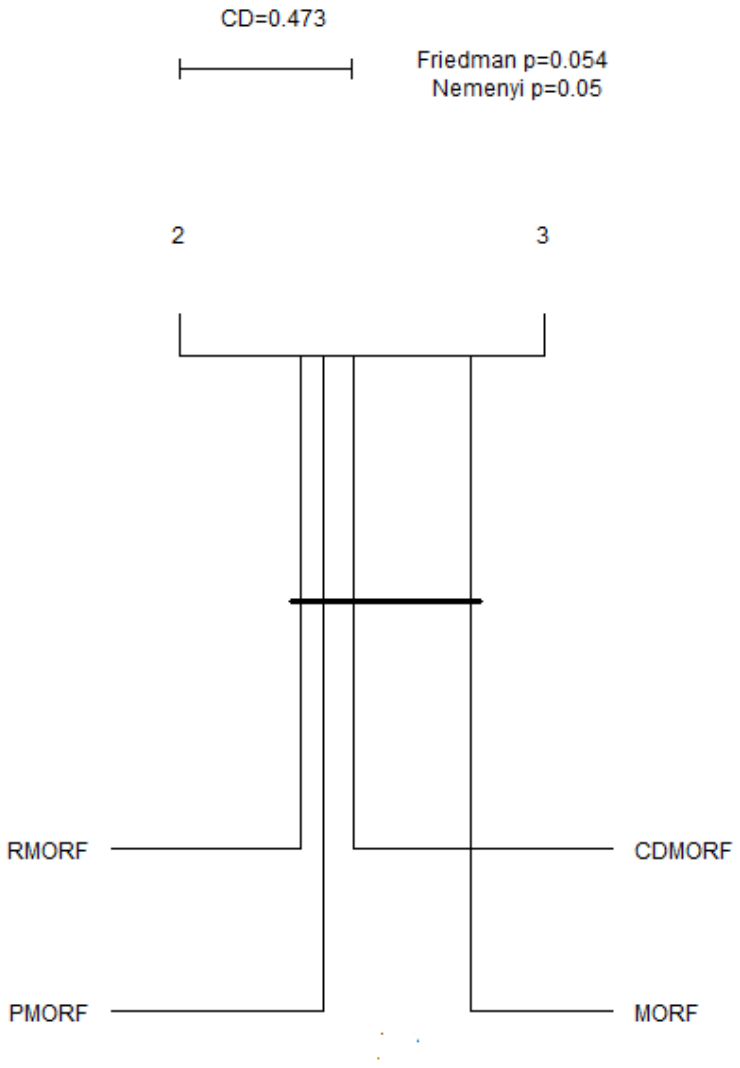

Figure 3.10. Per target multi-objective method comparisons using Nemenyi test.

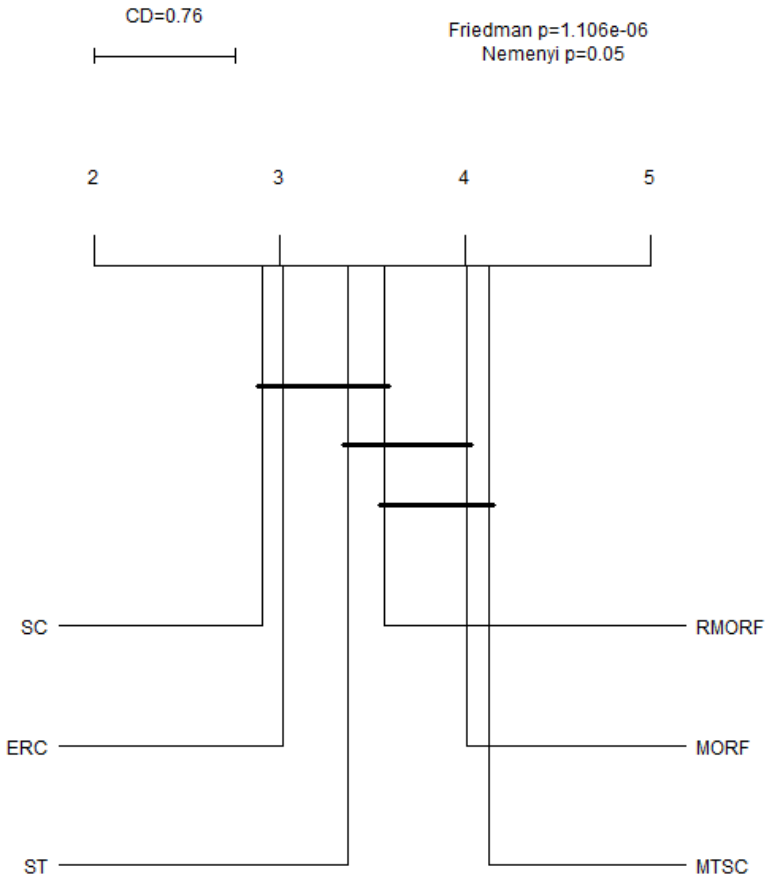

Figure 3.11. Per target method comparisons using Nemenyi test.

Considering the connected groups in Figure 3.11, we observe that SC results the lowest average rank and competitive with state of the art methods. In addition to that, Figure 3.12 represents a comparison of the errors and it also supports that, SC mostly makes equivalent or better predictions than ERC as most of the points scatter around or below of the line. In line with [5], our experiments verify that multi target approaches cannot significantly defeat single target versions. This is expected as the base regressor method is so powerful that it can take advantage of datasets' own information to a great extent. In addition, SC is significantly better than multi target methods, yet ERC and ST are not. In other words, a potent single target method can better utilize the target relations with a proper chaining design. Another remark on the importance of chain configuration reveals considering MTSC method.

Default MTSC flow actually complicates the learning process which is a sign of being cautious about transferring information.

Another observation can be made about the performance of MORF and PMORF methods. These methods also do not exhibit a significant difference in terms of performance. Keeping in mind that, MORF and RMORF methods are based on scalarization and Pareto approaches, respectively. With reference to the Figure 3.11, RMORF results superior performance than MORF.

In addition to its competitive performance, SC can aid to explain the way targets relate to each other. Moreover, sequence information in chain provide interesting inferences such as insights on dependency of the targets. Besides its predictive power, these chains could be used as tools for verification of prior information about a particular domain. This knowledge cannot be revealed by ERC therefore from practical point of view SC offers a valuable additional benefit besides its predictive power.

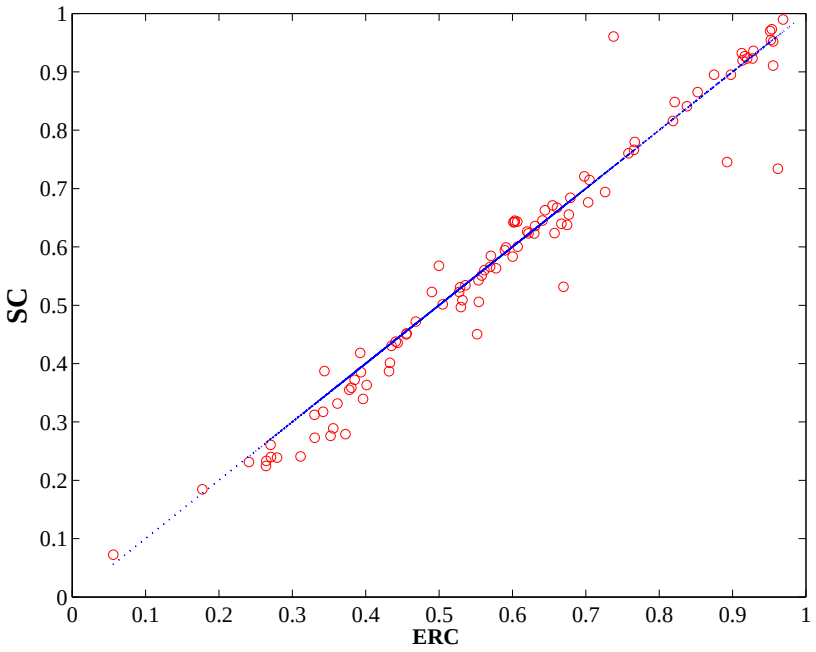

Figure 3.12. Per target errors of SC and ERC algorithms.

3.4.2.3. Per Dataset Analysis. And finally, we compare the method performances considering per dataset analysis (see Figure 3.13). Per dataset analysis requires averaging the prediction errors for each dataset. In other words, in per data set analysis we compared 18 observations which equivalent with total number of datasets used in the experiments. Half of the datasets have less than or equal to 3 targets and although there is no significant difference in terms of performances, another remark can be made by observing ERC, ST and SC are the methods with the lowest ranks. This situation also supports a better utilization potential of targets by using a suitable chaining strategy rather than multi task extensions of the selected classifier.

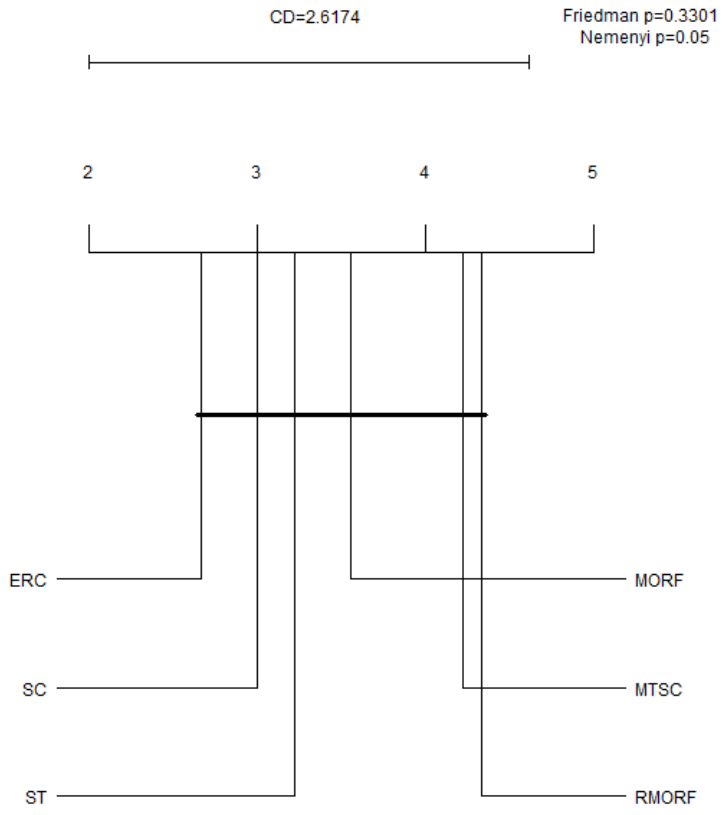

Figure 3.13. Method comparisons using Nemenyi test for per dataset.

3.4.2.4. Effect of Target Cardinality. In order to find out whether the number of targets are important on the effectiveness of the methods we grouped the datasets into two.

Considering the first group, we perform Nemenyi test for the datasets with number of targets less than or equal to 3 (see Figure 3.14). For such kind of datasets, ERC yields the lowest average rank however its performance is not significantly different from the proposed methods. ERC is good at in dealing datasets with less number of targets as a result of its chaining strategy since it covers the best chain configuration for sure.

For the datasets with greater number of targets, namely more than 4, SC obtains the lowest average ranks and is significantly better than multi target approaches. However, considering Figure 3.11 and 3.15, RMORF and ERC performances do not result in a significant difference which implies, RMORF is as good as ERC in terms of exploiting target information. As the number of targets grow, contribution of target interrelation as a side information become more apparent. In a sense, Figure 3.14 and Figure 3.15 support the effectiveness of use side information in learning algorithms by the comparison between SC and other algorithms.

3.4.2.5. Synthetic Experiments. We performed an experimental study in order to show whether multitask learning is a better practice than weighted sum or not. In this experiment we used a synthetically generated dataset with a heterogeneous target combination. We consider a multi-target dataset as heterogeneous if its target sets consist of categorical variables and continuous values, respectively. This specific problem structure allows us to explore the scaling effect on the learning process since entropy measures of a classification and a regression problem do not agree without scaling operation. To clarify, considering the classification task, range of the Gini entropy measure is simply $[0,\ 0.5]$. Likewise, for a regression task with mean squared error measure, the range is bounded by the continuous target itself. As a result, without scaling an equally weighted combination of the entropy measures is susceptible to be governed by the regression task. Or alternatively, considering a pure multi output regression problem where targets have different ranges, still the target/s with greater upper limit/s govern the learning process.

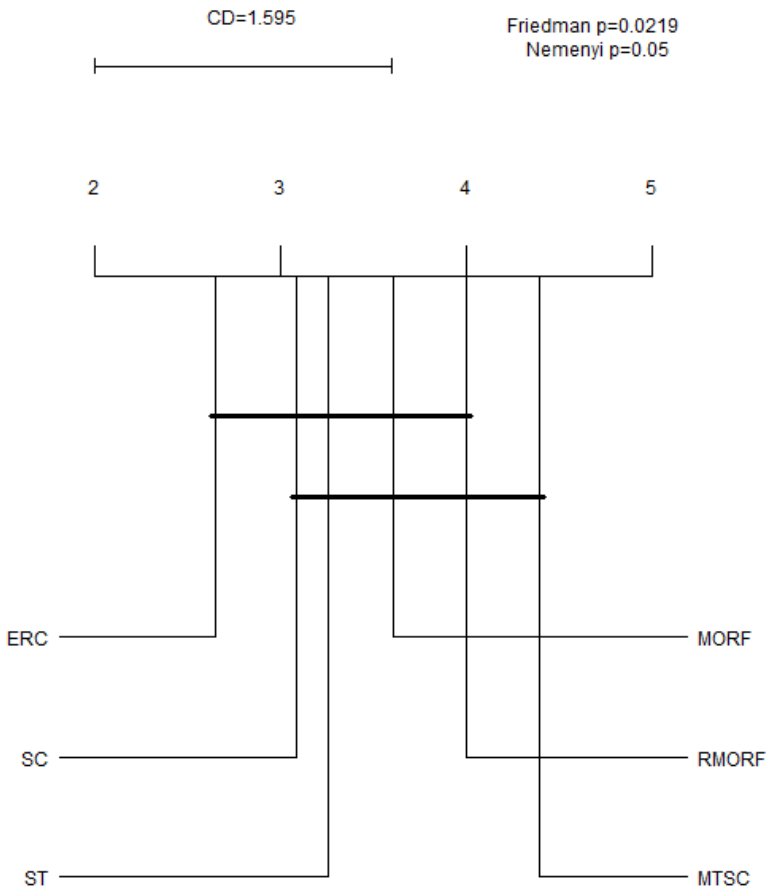

Figure 3.14. Method comparisons using Nemenyi test for datasets with number of targets less than or equal to 3.

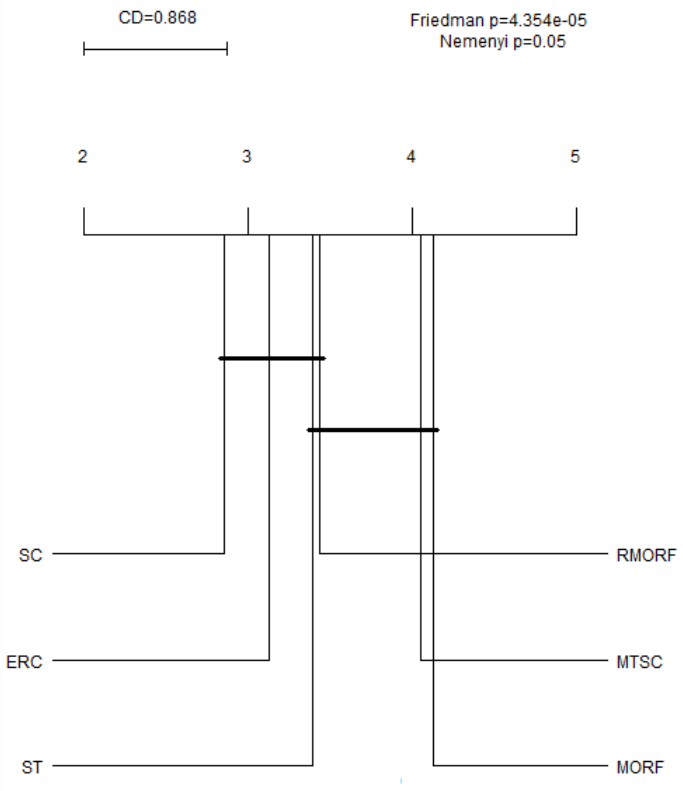

Figure 3.15. Method comparisons using Nemenyi test for datasets with number of targets greater than or equal to 4.

In addition to examination of the scaling effect, we generate the dataset by relating the targets with each other so that task relatedness would become an opportunity towards a better modelling practice (see Figure 3.16).

We prefer an XOR problem that is illustrated in Figure 3.16 as we want to test whether the given methods can benefit from task relatedness or not independent from type of task. The dataset contains two continuous features in addition to a mixture of different target types, i.e. one of the targets is categorical and the other one is numerical. In other words, the problem requires a classification and a regression task simultaneously. Figure 3.16 represents XOR dataset. We generate the XOR dataset as follows. We fix the covariances as an identity matrix and create four groups of multivariate normal random variables with means of $[5, 5], [15, 15], [5, 15], [15, 5]$. We used the random variables generated with the first two means and assign the result of the sum of x_1, x_2. In addition, we add a Gaussian noise to numerical target y_1 and assign a label A for the categorical target y_2. Likewise, we consider the random variable group created by using 3^{rd} and 4^{th} means and subtract x_2 from x_1. We repeat the Gaussian normal variable addition to the results and assign a label B for the categorical target y_2. The dataset can be found in [41].

Algorithms applied to XOR dataset are (i) RMORF, (ii) SC and (iii) equally weighted sum minimizer. Noting that, RMORF is a multitask method and it considers each target impurity in Pareto optimality sense throughout its derivation. In addition, SC is a single target method by its definition. And finally, we consider a tree ensemble that minimizes the sum of the Gini measure and a mean squared error that is scaled to $[0\ , 0.5]$. Table 3.2 illustrates ten-fold cross validation results in terms of the classification accuracy of y_1 and the RRMSE of y_2. Considering classification accuracy SC yields the best result however its regression quality is outperformed by its competitors. RMORF and Weighted Sum methods yield similar results in terms of both targets, yet Weighted Sum slightly outperforms RMORF for classification task and RMORF outperforms the rest for regression task. These results support that benefiting target relations changes the learning trajectory towards better.

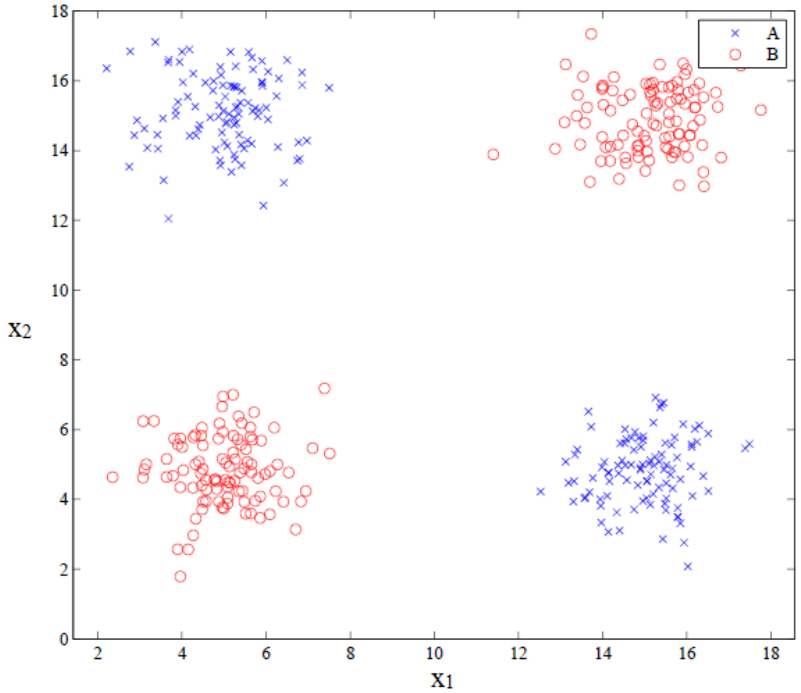

Figure 3.16. XOR dataset.

Table 3.2. Test results for XOR dataset (y_1: categorical, y_2: continuous).

Target	**RMORF**	**SC**	**Weighted Sum**
y_1	0.9300	0.9950	0.9475
y_2	0.0964	0.1207	0.0980

3.4.3. Computational Time Analysis

The aim of the analyses in this section is to check whether the theoretical estimations of algorithmic complexities are consistent with empirical results or not. We run our experiments by using a standard notebook with MS operating system and 16 GB RAM. The system contains four physical cores (i7-7700HQ, 2.8 GHz) and a single thread used throughout the experiments. We perform our experiments with *OES97* dataset since it has adequate number of targets, features and instances to observe the behaviour of the algorithms when they are subject to various settings.

We consider the following parameters p, m and n as the number of features, targets and instances, respectively. We set the experimental conditions based on the changes of these parameters. In each run we randomly select $\gamma \in \{0.2,\ 0.4,\ 0.6,\ 0.8,\ 1\}$ proportions for each of the parameters, and update the dataset accordingly. We collected results for updated datasets and represented the result with their average. All the values represent an average of five replications. And the ensemble size is fixed to 100 for each run. Observed training complexity of the SC and RMORF algorithms are given in Figures 3.17 and 3.19, respectively. Likewise, empirical test complexities of the algorithms are given in Figure 3.21 and Figure 3.22.

3.4.3.1. Empirical Training Complexity. Figure 3.17 and Figure 3.18 represent the time results based on γ proportions of N and M for learning with SC algorithm. In line with theoretical calculations, an increment in the number of targets implies a polynomial increase in corresponding time complexity for SC algorithm. On the other hand, Figure 3.18 illustrates the experiments based on γ proportions of N and p. Results given in Figure 3.18 supports the claim that SC algorithm is almost linear with number of features, namely p. Noting that, usually $p \gg m$ and a linear growth in number of features does not yield a polynomial increase in time complexity, hence we can conclude that SC algorithm is suitable for most of the dataset settings.

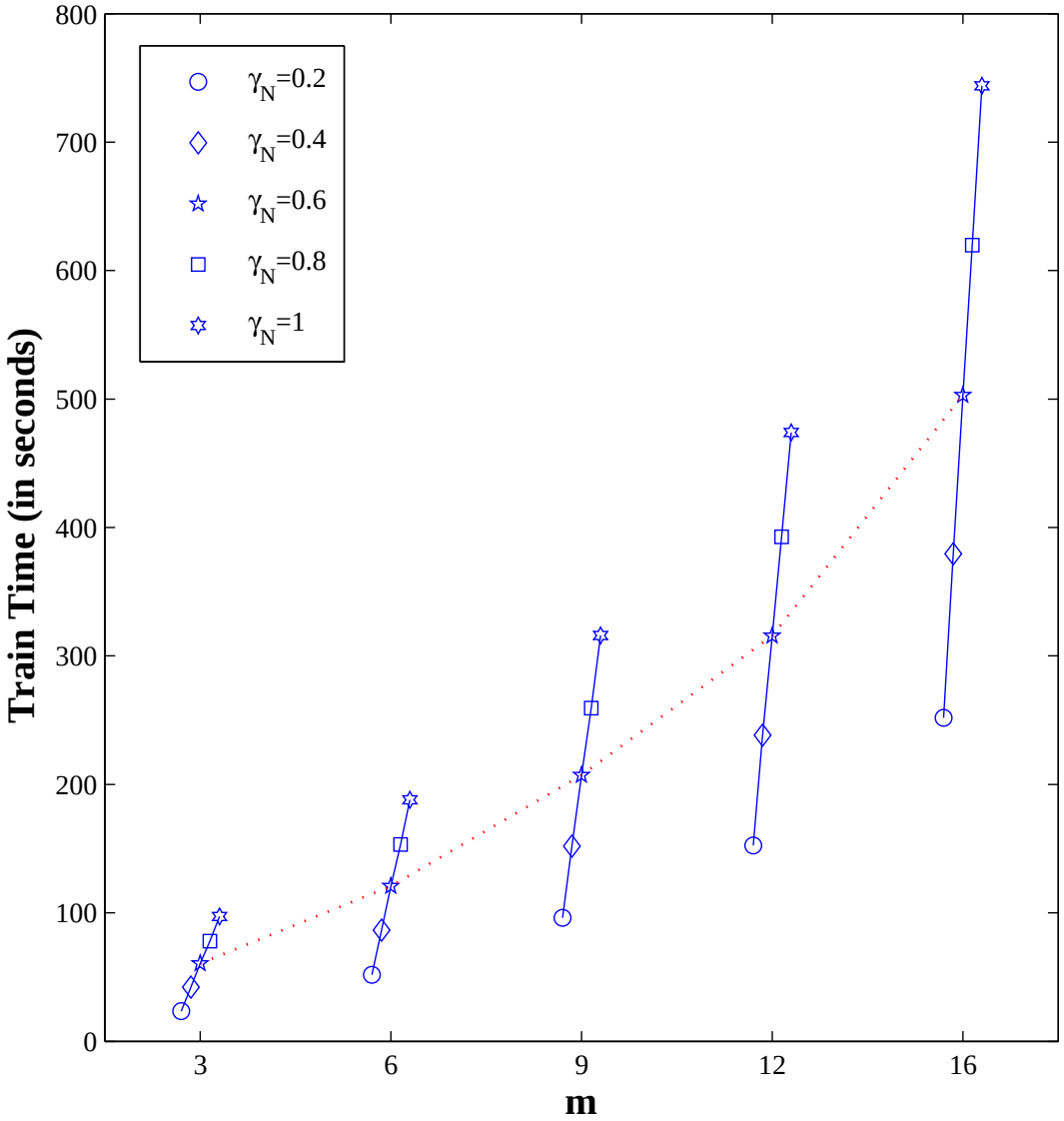

Figure 3.17. Empirical complexity results of training with SC algorithm for OES97 dataset with respect to number of targets.

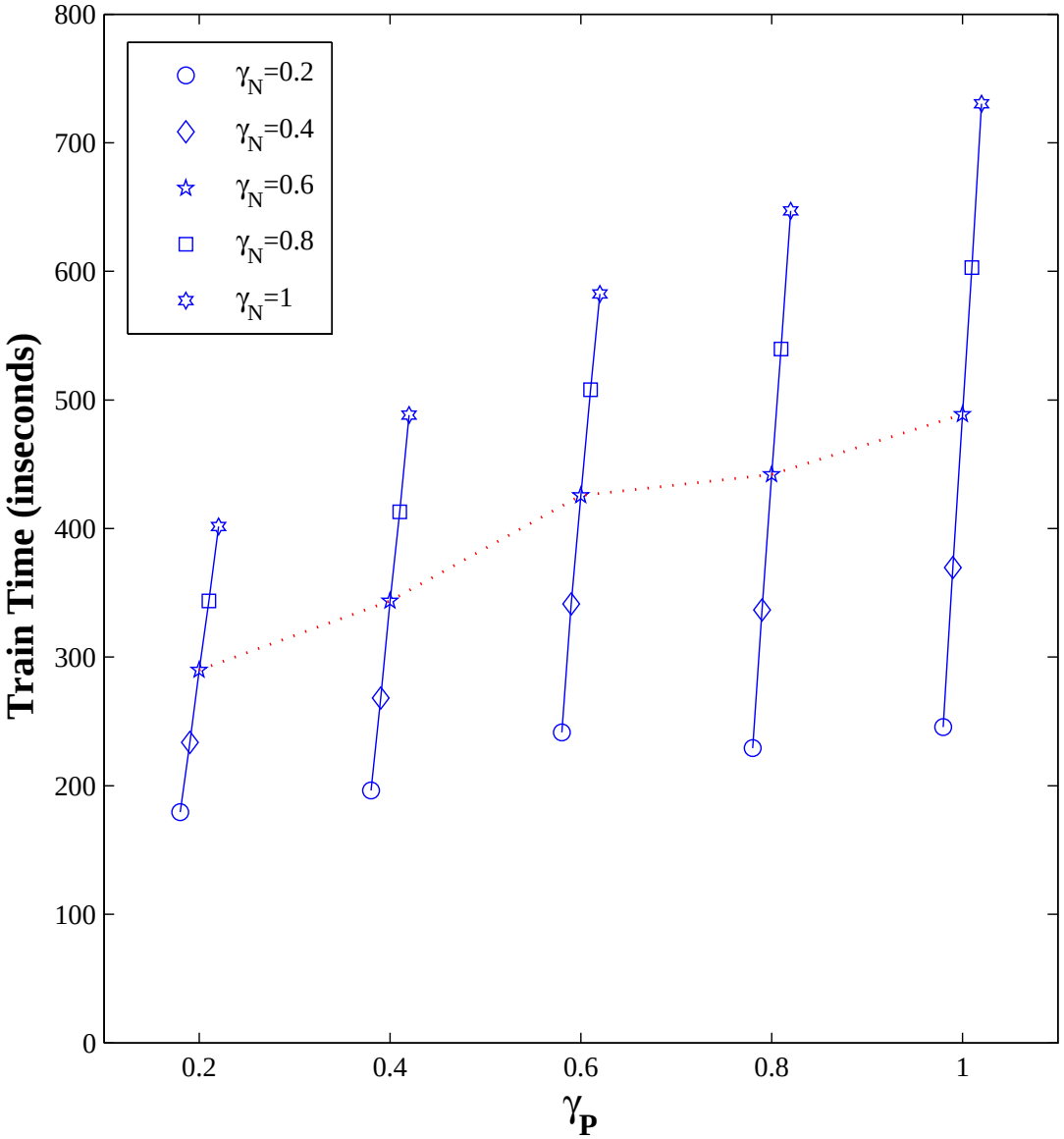

Figure 3.18. Empirical complexity results of training with SC algorithm for OES97 dataset with respect to feature proportions.

Figure 3.19 and Figure 3.20 illustrate the time results regarding to N and M data subsets for RMORF algorithm, and the observed linear increase is consistent with algorithmic complexity discussed in Section 4.3. In addition to that, Figure 3.20 represents the results for N and p varieties so that it supports the linear increase with increasing number of features and is consistent with theoretical algorithmic complexity.

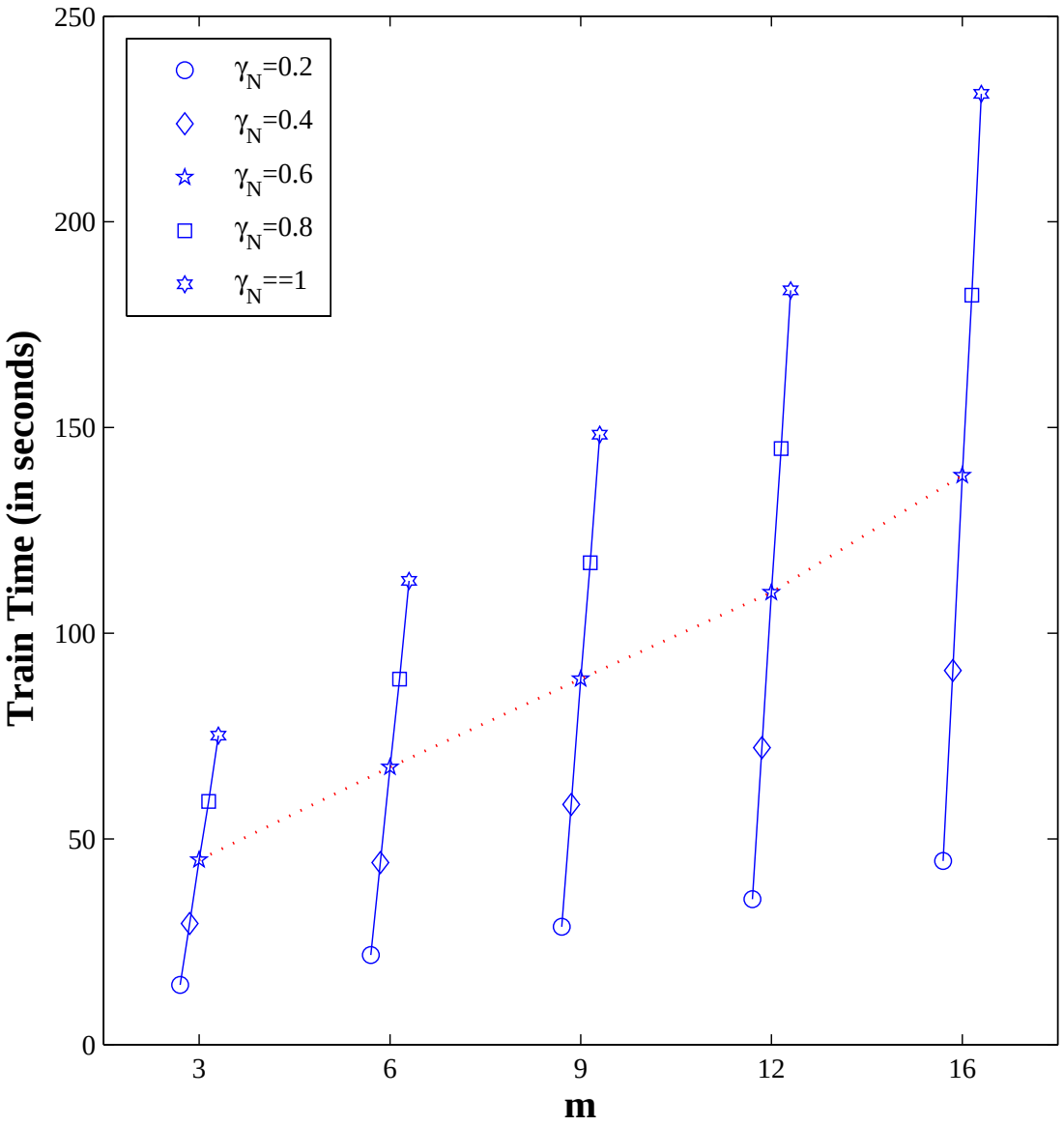

Figure 3.19. Empirical complexity results of training with RMORF algorithm for OES97 dataset with respect to number of targets.

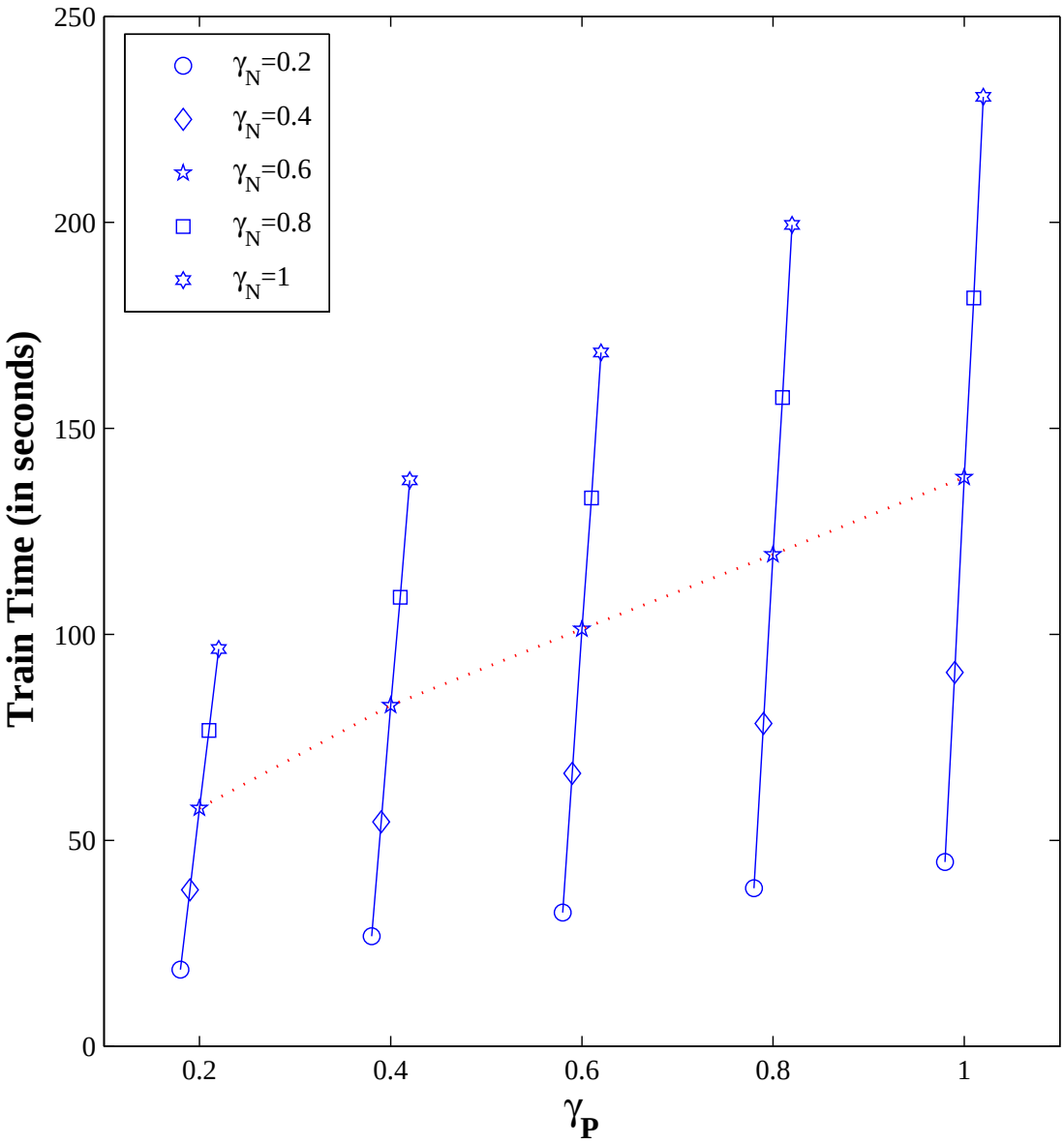

Figure 3.20. Empirical complexity results of training with RMORF algorithm for OES97 dataset with respect to feature proportions.

It is worth noting that time required to train SC is approximately four times larger than training RMRF under same conditions. In addition, we observe the superiority of global models over local models in terms of time complexity as RMORF is a global learner and SC is a combination of local learners.

3.4.3.2. Empirical Test Complexity. Test time refers to time elapsed in order to make prediction for a single instance. Considering test complexities, we specifically focused on effect of the targets as tree traverse for an instance during prediction requires relatively short time. Figure 3.21 shows that as the number of targets increase, it takes a longer time to make prediction for SC however this increment is in linear with number of targets. This is expected since it actually requires a gradual derivation of output. In other words, SC creates a sequence of local learners where length of the sequence is equal to number of targets.

In contrast to SC test time, RMORF prediction does not get worse as sharp as SC (see Figure 3.22). This is also not a surprising inference since RMORF produces multiple results in a-one-off manner. Combining these empirical evidences, our experiments support the benefit of using global learners when time is the major concern.

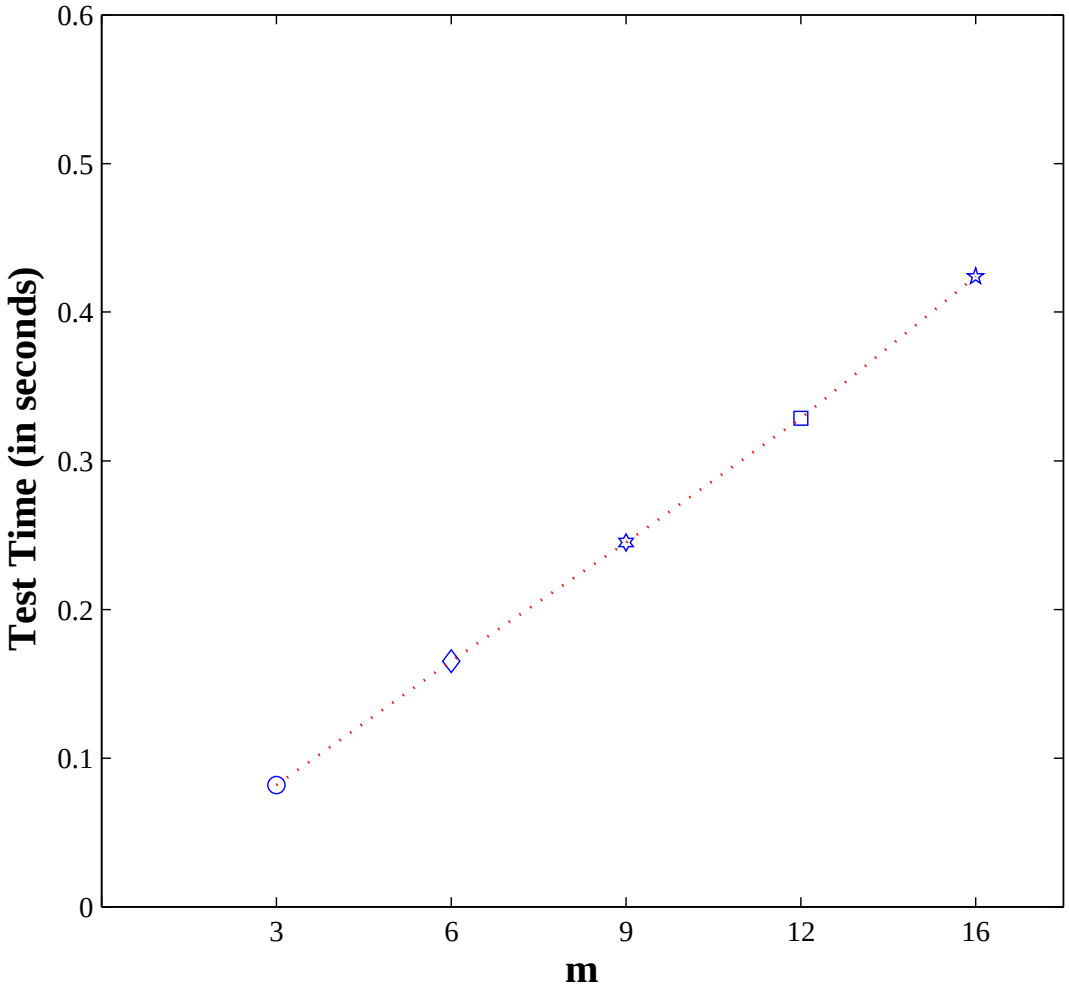

Figure 3.21. Empirical complexity results of testing with SC algorithm for OES97 dataset.

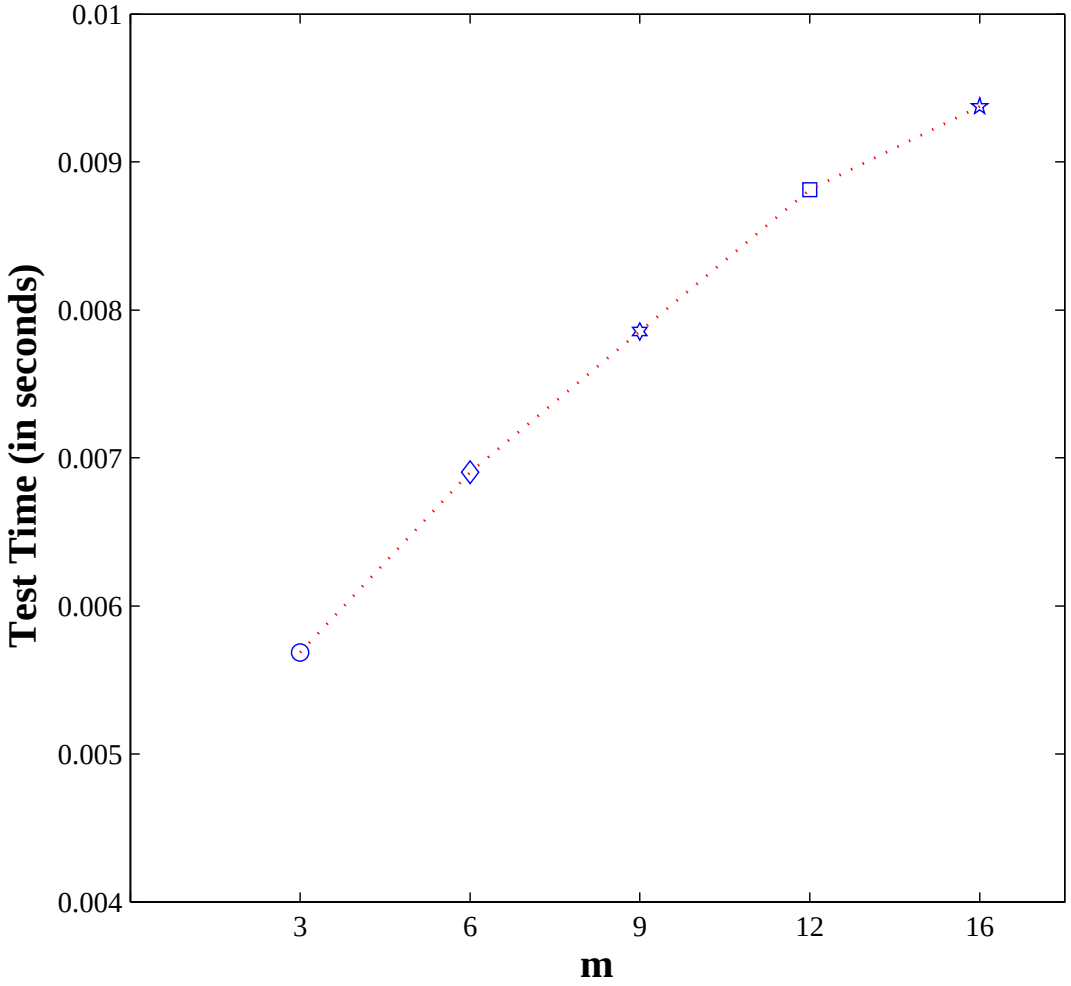

Figure 3.22. Empirical complexity results of testing with RMORF algorithm for OES97 dataset.

4. SEMI-SUPERVISED EXTENSIONS OF MULTITASK TREE ENSEMBLES

4.1. Introduction

Recent advances in hardware technologies and measurement techniques result in massive accumulation of anonymous data. However, there appears to be a consensus of opinion that manually labelling the data is a costly and challenging activity. For example, data identification may require experts with special skills or knowledge, i.e. in bioinformatics domain, or an abundant budget to evaluate specific designs of expensive experiments [43]. To overcome those issues, researchers have developed various strategies for gaining descriptive or predictive information. *Unsupervised learning* is an option to describe the patterns hidden in data and requires no label information. Besides unsupervised learning, predictive tasks could be totally based on label information, that is *supervised learning*. However, supervised learners can hardly provide reliable inferences when limited label information is available. Efforts on utilizing unlabelled data in predictive tasks result a hybrid of two learning strategies: *semi-supervised learning* (SSL).

Both supervised and unsupervised learning methods require optimization of specific score measures in model derivation. For supervised learning, those criteria could be sum of squared errors or impurities. For clustering, as a type of unsupervised learning, distance metrics or other measures quantifying similarity or dissimilarity among instances might be employed. From SSL perspective, integrating information from supervised and unsupervised sources is a strategy for better utilization of all the data on hand when it meets *i*) *smoothness*, *ii*) *clustering* and *iii*) *manifold* assumptions [13,18]. To clarify, smoothness assumption results similar labels for instances with similar feature information. Clustering assumption aids to benefiting from setting decision boundaries passing through low density regions. Manifold assumption allows to find lower dimensional representations of higher dimensional data.

In this study, our SSL methodology relies on smoothness and clustering assumptions. Since the labelled and unlabelled parts share the similar distributional properties, we aim to exploit abundant unlabelled part to discover statistical and geometric properties to improve predictive performance for multi-target data sets [15].

To illustrate our motivation, we refer to the example in Figure 4.1. In this example, we generated a data set that consists of a single feature (X) and a single target (Y). 50 instances are drawn from two Gaussian distributions and they are shown on Figure 4.1. Four instances are randomly selected to represent the labelled data and the remaining part are left as unlabelled. Under these conditions, supervised and semi-supervised strategies for decision trees result in two different splits. Noting that, supervised strategy merely considers the labelled instances as available information, however semi-supervised approach enriches its selection by taking into account the unsupervised knowledge, that is the distributional property of unlabelled part. When predictive performances of these two models are compared, semi-supervised strategy helps to enhance the performance of its supervised counterpart by shifting the decision boundary towards the void area between data groups. This inference hints for a better predictive performance for semi-supervised strategy against the supervised one. Despite the proven power of SSL, there are various pitfalls that need special attention in design of it.

A major shortcoming about distance measures is that they lose their functionality in higher dimensions [12]. Obviously, alternative information types other than traditional distances to measure the affinities (or similarities) between instances are essential to handle high dimensional data. Moreover, targets and/or features could be in a mixed form of categorical, ordinal and continuous values, and distance notion is not well defined for mixed type of values. As a consequence, traditional distance based calculations are susceptible to get easily complicated by types of feature or target values.

In SSL, though sequential use of supervised and unsupervised modules does not pose problem, their simultaneous usage deserves careful handling.

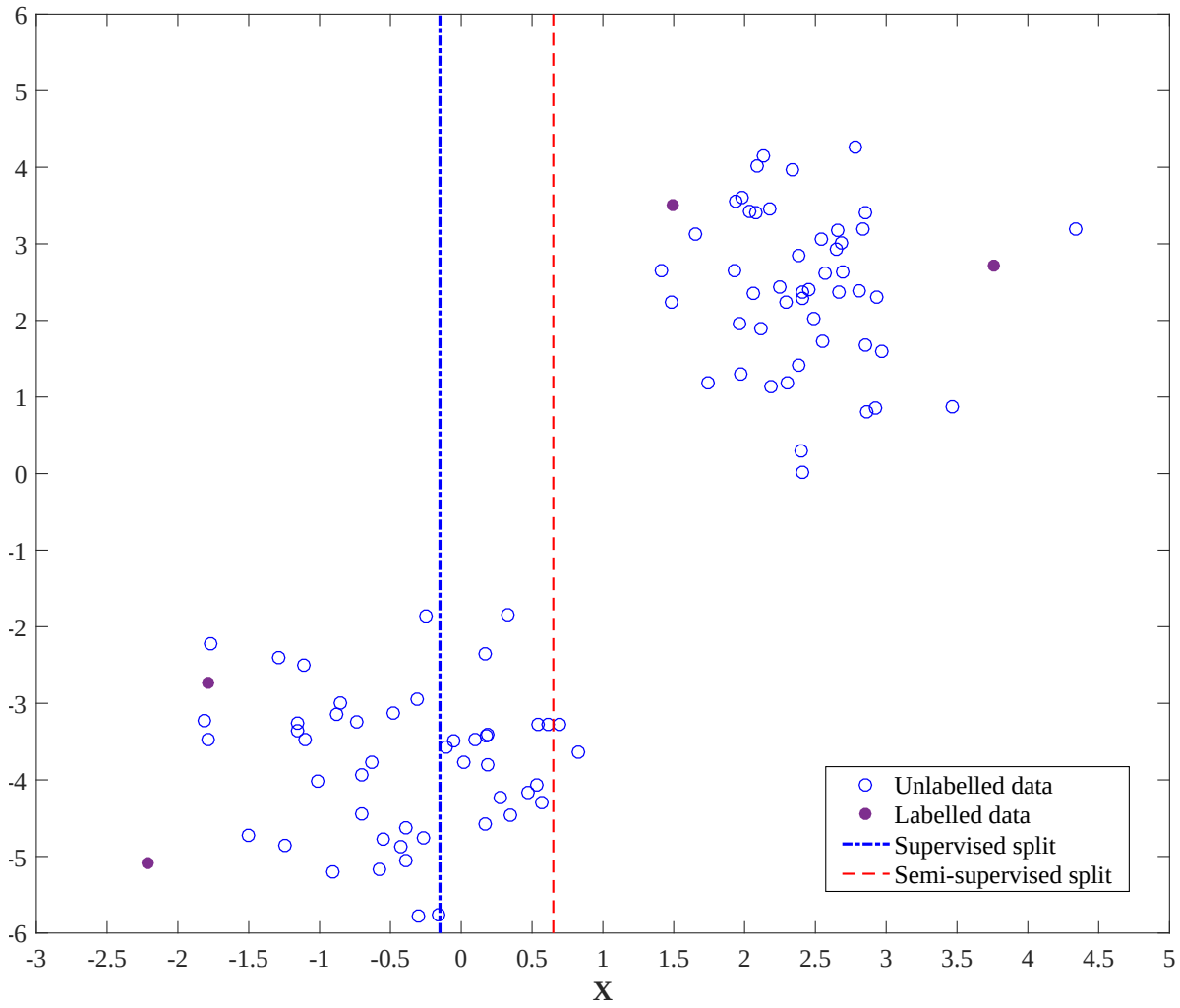

Figure 4.1. Splits generated by semi-supervised and supervised trees.

To clarify scales of supervised (e.g., sum of squared errors) and unsupervised (e.g., distance between two instances) learning criteria which are used in model derivation, do not need to agree. Data transformation may aid to handle those issues yet it is undesirable due to several reasons. The best transformation methods that fits the data cannot be known in advance, hence it requires additional effort to reveal the optimum technique [44]. In addition, it may risk of changing the statistical properties of the data [12]. To exemplify, discretization of continuous values may result in loss of information or converting country or gender type of information into numbers does not result comparable outputs as before. Even some type of linear transformations do not harm the structural properties of the values, they fail to handle values that are out of range [45]. Yet another issue with data transformation stems from omitting the similarity informations hidden in original values and as a consequence structural characteristics of the data sets cannot be preserved [46]. To conclude, handling scale differences is a worthwhile challenge to consider in learning problems.

Although existing SSL methods are well-established, they can be criticized as their design allows to model single target data sets [16]. Very few SSL studies can be found for multi-target data sets nevertheless they can only handle either homogeneously categorical [17, 19], or continuous targets [16, 20, 47]. Aforementioned approaches can be criticized as they train the learning model based on a weighted sum form as a combination of normalized score measures separately.

From side information point of view, exploiting the target relations via *multi-task learning* (MTL) rather than individual calculations, may reinforce the model's prediction quality [4]. Unlabelled part of the data can be considered as another source of side information as it provides insights about the data in addition to labelled part. We argue that, combining multi-task models in semi-supervised setting assists for a better learning trajectory specifically when access to the labelled data is limited.

As an instance of multi-task learning model, multi-objective adaptation of decision trees allows to integrate information for multiple tasks without being challenged by scaling issues.

Decision trees are known to be handy as they can handle high dimensional data and allows to create similarity measures based on the possible partitions. In addition, they do not make any assumption about the underlying structure of the data. However a single decision tree may not be enough to explore and exploit the most of the data [48]. Besides, one single classifier may not be enough to capture several local distributions since in higher dimensions cluster structures may not be detectable and scanning all possible feature combinations is not feasible [19]. Moreover, a single classifier may not result in stable models and may be sensitive to parametrizations. *Ensemble learning* is proven to be a working alternative in addressing those problems [49].

Similarity information between instances is an important by-product of tree ensembles and constitutes the linchpin of our study. We propose to derive the unsupervised criteria by using the similarity information delivered by total random forests [27, 29]. By doing so, we aim to explore the unsupervised information without being challenged by features with scale differences. In addition, considering data sets with large number features similarity information provides more meaningful comparisons than classical norm type distances at higher dimensions.

In this study, we propose to extend multi-objective supervised learning strategy to its semi-supervised version by including the clustering criteria to the supervised criteria set in an ensemble setting.

Our experimental results show that proposed approaches are promising in semi-supervised applications without being challenged by scale differences either in targets or features.

The rest of Chapter 4 is organized as follows: Section 4.2 provides the relevant literature and positions our study in the field. Section 4.3 introduces the proposed semi-supervised methods. We report the experimental results in Section 4.4 and discuss the findings.

4.2. Related Work

This section focuses on semi-supervised extensions of multi-target tree ensembles, hence the scope of literature review is limited to SSL applications of tree learners. Decision trees are often used for supervised learning tasks, however handling unsupervised tasks is possible with minor modifications. To clarify, in training of classical supervised trees target related loss functions are optimized. Since there exists no target information in unsupervised tasks, error functions should be replaced with some criteria which measure the partition quality from unsupervised point of view. Those functional forms could be used to measure either within cluster variation [50] or some probabilistic entropies [2] in the partitions. Noting that supervised trees can be used for unsupervised tasks yet those approaches are beyond scope of this paper [51].

As an unsupervised learner, clustering trees assume the data as a hierarchy of clusters. As an earlier attempt, study given in [50] constructs trees by reducing intra-cluster variation at each split without considering target influence. In [52], trees are created via a feature homogeneity evaluation procedure unlike classical target based losses. In [53], attributes contribute the tree construction in addition to target guidance. Their study considers multiple classification tasks and combines two kind of information in a weighted sum form. Their work considers supervised setting for classification tasks and its performance is not well defined for data sets with scarce label information.

Semi-supervised RFs are used for co-training purpose in [54]. Co-training is a SSL technique where separate learners assists to increase the labelled instances by relying their most confident predictions. Another semi-supervised RF study can be found in [55]. In this study a compact but non-convex loss function is defined for the node splits. In order to solve the optimization problem, a deterministic simulated annealing heuristic is used therein. Study given in [55] is criticized as the classifier is susceptible to overfit and it does not provide robust results. This issue stems from its solution procedure since it suffers from being sensitive to inefficient initialization and getting captured to local maxima [56].

Another semi-supervised RF construction procedure is given in [56]. They evaluate split quality by kernel based density estimation function. This study is valid for data sets with homogeneously numerical features hence cannot be applied for mixed type of data sets. Moreover, they consider a classification problem for single target setting which cannot be effectively used for multi target problems. Study given in [57] is a more recent SSL application for single target classification problem. This study augments its labelled data pool with self training and uses standard decision trees as base learners. They train trees iteratively and at each iteration the tree enlarges its training set with its own most confident predictions.

Utilizing multi-objective structures is another approach in modelling SSL problems. One of multi-objective SSL attempt is given [58]. In this study language modelling problem is considered as a sum of two Bayesian risk measure. They use ε-constraint method to solve their model. In [59] a self-training based RF is used for multiple target regression problem. Their base learners are predictive clustering trees given in [50] where they do not consider target information in classifier derivation. They evaluate the reliability of label predictions with an approximation of variance of errors and select the most confident labels to augment the labelled data pool.

In more recent studies, multi-objective predictive clustering trees are given in [16] and [17]. In those works, the authors design the split function as a weighted sum of supervised and unsupervised impurities for multi target regression and classification problems, respectively. In [16], authors extend the classical predictive clustering trees (PCT) into their semi-supervised version by including the target contribution in tree derivation. Classical PCTs handle supervised learning tasks and in training PCTs a score measure based on reduction of target variances is used. Here, individual changes of the variances are calculated and an overall difference is obtained by averaging those values. Split value with the largest variance reduction on the average is selected as the best partition and it is applied to obtain child nodes. Authors update supervised PCT split score to its semi-supervised version by adding a weighted term to control the change in terms of features in parallel with targets.

Reduction in average of individual variance and Gini values are considered for numerical and categorical features, respectively.

Our study departs from studies given in [16, 17] in different aspects. First of all, our aim is to avoid setting of weight parameters as they introduce additional processing overhead. Clearly, classifier performance depends on accurate estimation of those parameters which may not be an easy task for users. Moreover, those studies can handle pure classification or regression problems. Our second aim is to derive a SSL strategy so that it can handle mixed type of targets simultaneously. Third, in those studies features individually contribute in terms of their variances and impurities for numerical and categorical data, respectively. We consider cluster homogeneities and the goodness of separations simultaneously. By doing so, our strategy complies with clustering assumption and resembles to margin maximization approaches and serves towards better partitions by favouring less dense areas in a tractable fashion. In addition, not to transform either data itself or the values appear in split function is our another concern.

4.3. Methods

4.3.1. Semi-supervised Multitask Random Forests

In this section we introduce SSL algorithms designed for decision trees. We will describe the modified split function and proposed labelling policy as well. We develop this method to simultaneously handle mixed type of inputs and outputs for multi-task tree ensembles in semi-supervised setting. We employ multi-objective adaptation of supervised trees so that we can seamlessly extend the split function with unsupervised information inclusion. Supervised score measure used in multi-objective trees consists of t values. Each value returns a measure for partition quality in terms of sum of squared error (SSE) or Gini index for continuous and categorical targets, respectively.

Assuming a child node $r \in \{left, right\}$, SSE_r^j is the sum of squared differences of target value (y_{ij}) and the mean response ($\hat{y}_{ij}$) of instances contained thereof:

$$SSE_r = \sum_{i \in node_r} (y_{ij} - \hat{y}_{ij})^2 \tag{4.1}$$

For categorical targets, Gini index G_r^j for a node r assesses the node impurities by using relative class frequencies, p_{rc}, $r \in \{left, right\}$ and $c \in C$:

$$G_r^j = \sum_{c \in C^j} (p_{rc}(1 - p_{rc})) \tag{4.2}$$

Values in the middle of each consecutive observation selected for split performance evaluation. By using each split candidate, SSE and Gini indices are calculated and combined in a vector form, that is Ω^s. Assuming first s of m targets are categorical and remaining targets are continuous, then Ω^s vector for each split candidate is represented with Equation (4.3):

$$\Omega^s = \{G^1, \ldots, G^s, SSE^{s+1}, \ldots, SSE^m\} \tag{4.3}$$

In order to convert a completely supervised score to a semi-supervised one, we combine Ω^s with unsupervised scores (Ω^u). We assume each child node as a cluster, hence cluster quality indicators are included in split evaluation. We employ two different unsupervised measure set.

4.3.1.1. Unsupervised Scores with Euclidean Distance. Ω^u criteria consist of *i*) distance between node medoids, *ii*) information gain based on Shannon entropy (see Equation (2.5)), and *iii*) distance between nodes. *Criterion i* aids to keep the intense areas of the clusters as separate as possible. *Criterion ii* aims to provide a measure for cluster compactness [2]. *Criterion iii* acts like a margin score similar to the min-linkage in derivation of hierarchical clustering. In other words, *Criterion iii* refers to the distance between the closest instances from two data groups (in this case nodes).

For each partition these values are calculated and Ω^u part of the overall score measure is calculated. Noting that, these calculations are valid when the features are continuous. In order to handle this issue, we replace Euclidean calculations with dissimilarity measures as an alternative. Henceforth, we name the semi-supervised algorithm based on Euclidean distance as Euclidean random forest (ERF).

4.3.1.2. Unsupervised Scores with Dissimilarities. Similarity information of the instances are derived by using total random forests before creating the semi-supervised model [29]. Following that, similarity information is divided by number of trees and then converted to dissimilarity information by using the relation proposed by [27] (See Equation (4.4)).

$$DIS_{ij} = \sqrt{1 - SIM_{ij}} \tag{4.4}$$

Concerns and structures of *criteria i* and *iii* remain the same yet we consider the dissimilarities rather than Euclidean distance in their derivation. Since dissimilarity measure is not challenged by scale issues, we consider to minimize the deviations from cluster medoids similar to sum of squared error calculations as an alternative for cluster compactness. Figure 4.2 illustrates a toy example for dissimilarity calculations. Assuming a group of three trees are created for a dataset with 8 instances. Rectangles and ellipticals refer to root or intermediate nodes and leaves, respectively. We denote each instance on the leaf where it falls. To exemplify, considering the tree on the leftmost of Figure 4.2, instance 1 and 2 terminates at the same node. Visit counts matrix represents number of times meet at the same leaf among three trees. Proximity matrix is obtained by simply dividing by three each element appears in $Visit\ count$ matrix. And finally, by using the relation given in Equation (4.4), we obtain well-known $Dissimilarity$ matrix presented by [27].

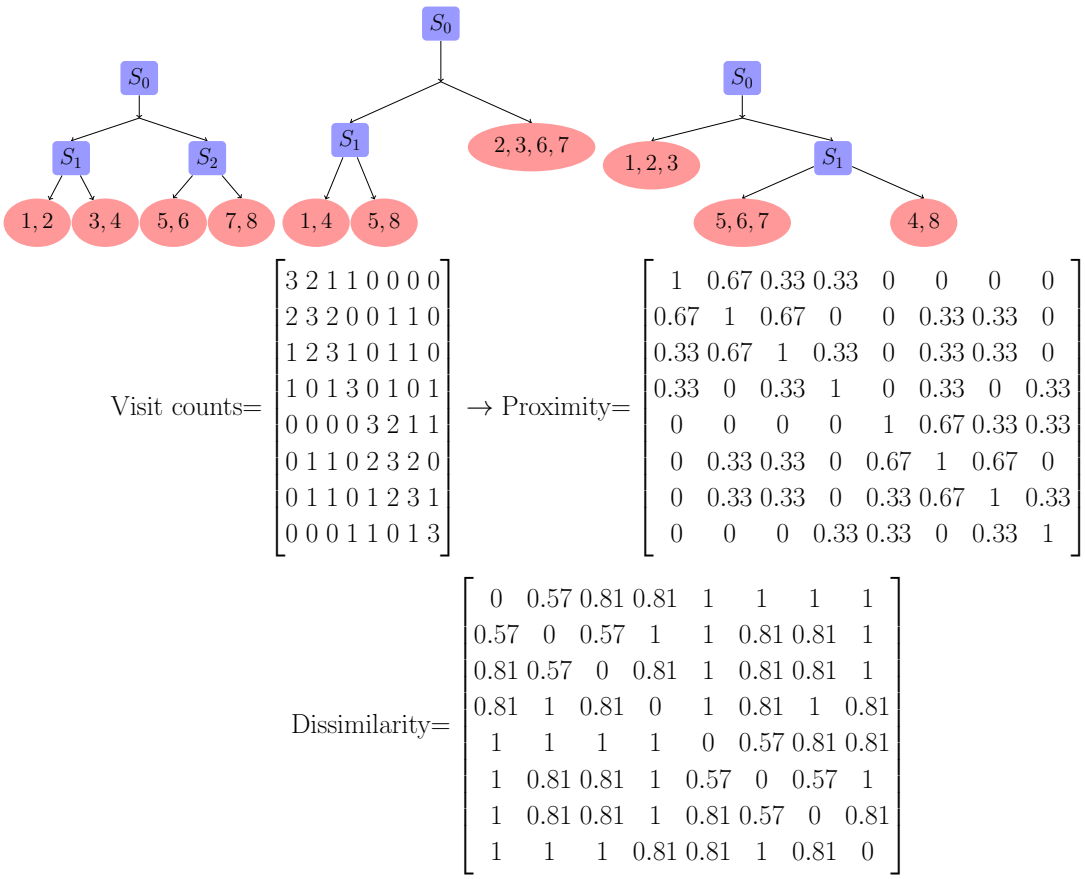

Figure 4.2. A toy example for dissimilarity matrix calculations.

Once we calculate unsupervised criteria Ω^u, we combine them with Ω^s and obtain semi-supervised split score $\Omega = \{\Omega^s, \Omega^u\}$. We represent the semi-supervised algorithm based on total random forest proximities with TRF.

Despite its several merits, multi-criteria comparison may suffer from curse of dimensionality. Proposed design is susceptible to the given phenomena when the number of targets inflates. For example, a data set with 16 targets is associated with supervised criteria vector Ω^s consists of 16 values. To cope with this issue, we provide an alternative method based on random selection of the targets in node split evaluation. To clarify, in TRF all the targets contribute to the supervised learning criteria Ω^s. In the alternative version, we randomly select three targets (if available), and consider their individual SSE values.

In other words, if the number of targets less than or equal to three, procedure uses all the targets' SSE values, otherwise it uses a random three of those values. Henceforth this strategy is called Random TRF (R-TRF). Noting that, at each split selection, this approach implicitly assigns zero weights to unselected targets.

4.3.1.3. Training. Training procedure is based on the algorithm given in Figure 4.3 with modified vector of split scores Ω. Combination of supervised and unsupervised criteria results the overall score measure. For each criterion, values are arranged in ascending order and they are associated with their ranks. Candidate with the lowest average rank is selected and child nodes are created recursively until a stopping criterion is met.

Require Training set D with n instances, stopping conditions
if termination criterion **then**
  Create leaf node and label it with its target averages;
  Return leaf node;
else
  Draw a P^* random subset of the input variables;
  Examine all possible binary splits for each of the input subset P^*;
  Create the set of first frontier with respect to the quality measure in vector form;
  Select the split with minimum average rank;
  Apply the best split and create left node and right node;
  Return Recursive Partitioning(left node);
  Return Recursive Partitioning(right node);
end if

Figure 4.3. Rank Based Semi-Supervised Tree Algorithm.

4.3.1.4. Prediction. Another key point of a SSL algorithm is *prediction policy* employed therein. In order to obtain a prediction for an unlabelled instance, it is enforced to traverse the tree until it visits a leaf node. Tree traversal may terminate in two different type of leaves:

- *Case 1:* Some of the instances in a leaf have their own labels.
- *Case 2:* All instances in a leaf are unlabelled.

We directly associate y_i target values of originally labelled instance x_i, to the leaf that belongs to (*Case 1* leaf). Hence, any instance terminates at a *Case 1* leaf is associated with available target information. Since *Case 2* leaves do not have any labelled instance, those leaves inherit the label information of its parent node.

4.3.2. Algorithmic Complexity

Computational complexity of a standard decision is $\mathcal{O}(PN \log N)$ where P and N refer to number of features and instances, respectively. In our models we use $\sqrt{P}$ features in split evaluation. In addition, we consider each of the target one by one, hence complexity of computations at each split becomes multiple of number of targets, that is M. Since we also need to consider an additional U number of unsupervised criteria at each split, overall computational complexity of a single TRF tree $\mathcal{O}((M + U)\sqrt{P}N \log N)$. Assuming ensemble size is set to J, complexity of TRF is concluded as $\mathcal{O}(J(M + U)\sqrt{P}N \log N)$.

4.4. Experiments and Results

4.4.1. Experiments

Data set properties are given in Table 4.1 and they are available in website given in work of [5]. We compared our algorithms with semi-supervised PCTs presented in [16] as the software is accessible and their results provide competitive benchmarks. We refer the reader to details of the algorithm given in [16].

Following the setting given [16], we consider the performance of a single PCT and PCT ensembles (PCTF). Benchmark algorithm is implemented in WEKA and can be found in website provided in [16]. In line with their study, 100 bagged trees are used with the same parametrizations given therein. We propose multi-task semi-supervised forests with different unsupervised information based on i) Euclidean distances and ii) dissimilarities between instances. We employ ten-fold-cross validation with an ensemble size of 100 trees. We define stopping condition by setting maximum number of labelled instances to one. We implement the methods in MATLAB 2016A. For similarity calculations we use a ready implementation of total random forests of 100 trees in Python environment with default parametrizations [29, 60]. All the experiments were performed by using a standard notebook with MS operating system and 16 GB RAM. The system contains four physical cores (i7-7700HQ, 2.8 GHz) and a single thread used throughout the experiments.

Code and data files are shared in a github project for the sake of reproducibility [41]. In addition, for fair comparison we use the same training and testing instance sets for all methods.

Table 4.1. Datasets for semi-supervised experiments.

Dataset	**Samples** (N)	**Features** (p)	**Targets** (m)
ENB	763	8	2
Water Quality	1060	16	14
OES10	403	298	16
OES97	334	263	16
CalHouse	1032	7	2
MP5	700	80	4
MP6	700	80	4

We construct the set of labelled instances by randomly selecting 5%, 10% and 20% of the data and use the remaining as unlabelled. Following the scenarios given in [16], we evaluate the performances for two settings.

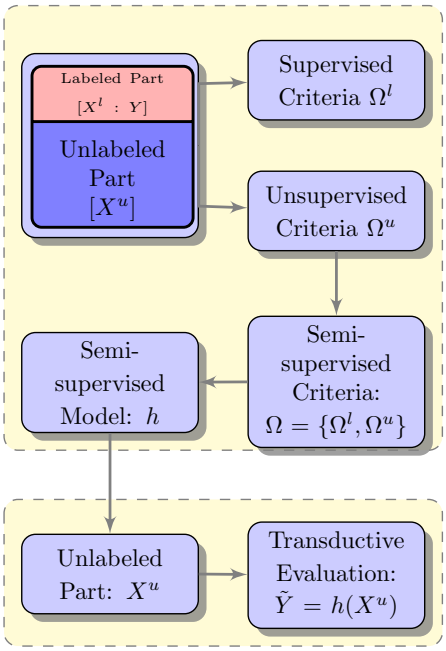

Figure 4.4. Transductive evaluation illustration.

First experiment series are designed for *tranductive evaluations* where all the unlabelled data is used in training and the performances are reported based on those data which are previously seen in model derivation (see Figure 4.4). Second experimental set-up is designed for *inductive evaluation* where 10% of the labelled data initially is kept apart from learning phase and the performance results are collected based on those unseen part (see Figure 4.5). Noting that, those evaluation settings are not directly comparable as their training and testing sets are different that each other. In addition, performances results are based on previously seen examples, hence inevitably results are biased in transductive evaluation.

4.4.2. Results and Discussions

We want to compare performances of multiple algorithms for multiple data sets hence we use the statistical methods proposed in [42]. In order to understand whether the difference in performances of the algorithms is statistically significant we employ Friedman test which is a non-parametric version of ANOVA. Following that, we use Nemenyi test to find out the whether an algorithm significantly outperforms the others.

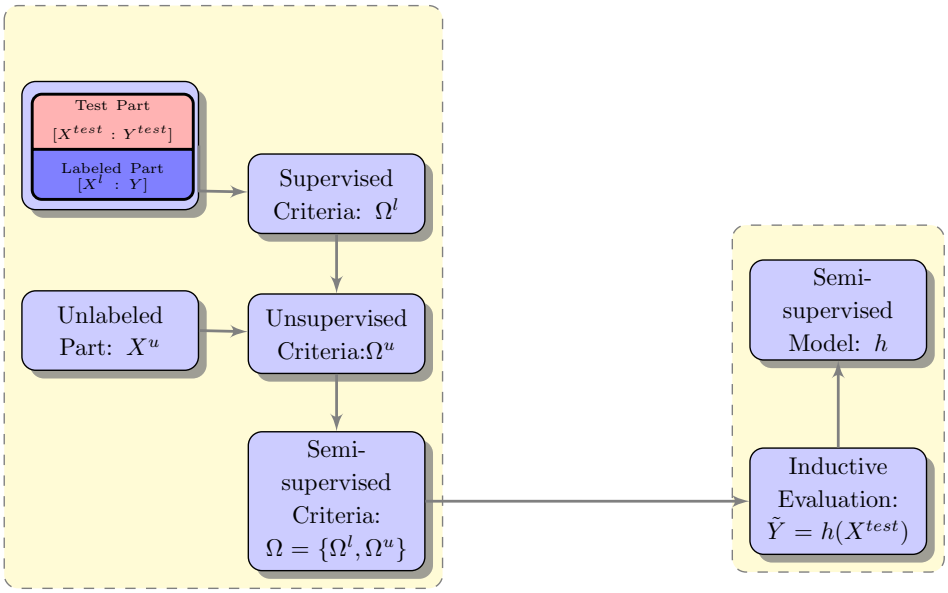

Figure 4.5. Inductive evaluation illustration.

Nemenyi test represents average ranks with a scale and on this scale algorithms with no statistical difference in terms of predictive performance are connected each other.

4.4.2.1. Out-of-Bag Analysis. In order to understand the effect of semi-supervised strategy from feature and target perspectives, we analyse out-of-bag (OOB) error curves. To calculate OOB error for a sample, predictions are taken from trees which have not seen that sample before in its training. We use a data set with eight features and two targets (ENB data set) for OOB analysis. Both of two targets and five out of eight features are continuous, and three of them are ordinal. In OOB experiments we always set proportion of the labelled instances to 10%. In Figure 4.6, RRMSE values for each target are given on the left. As it is expected, RRMSE curve for each target decreases and tends to saturate as the size of the ensemble grows. The results are consistent with the inversely proportional relation between OOB error and ensemble.

In Figure 4.6, sum of squared error (SSE) for features are given on the right. Similar to estimations made for targets, we calculate the sum of squared errors for features. Our purpose is to understand whether the semi-supervised model captures the distributional properties of the data or not.

Apart from ordinal features, Figure 4.6 shows that deviations from corresponding average value decrease following a peak point. That behaviour supports model's ability of capturing the distributional properties of the data hence it suggests about the strength and contribution potential of the unsupervised criteria to the learning process. Predictive performances are reported with Relative Root Mean Squared Error (RRMSE). RRMSE of a learning model h_j is given in Equation (3.1)

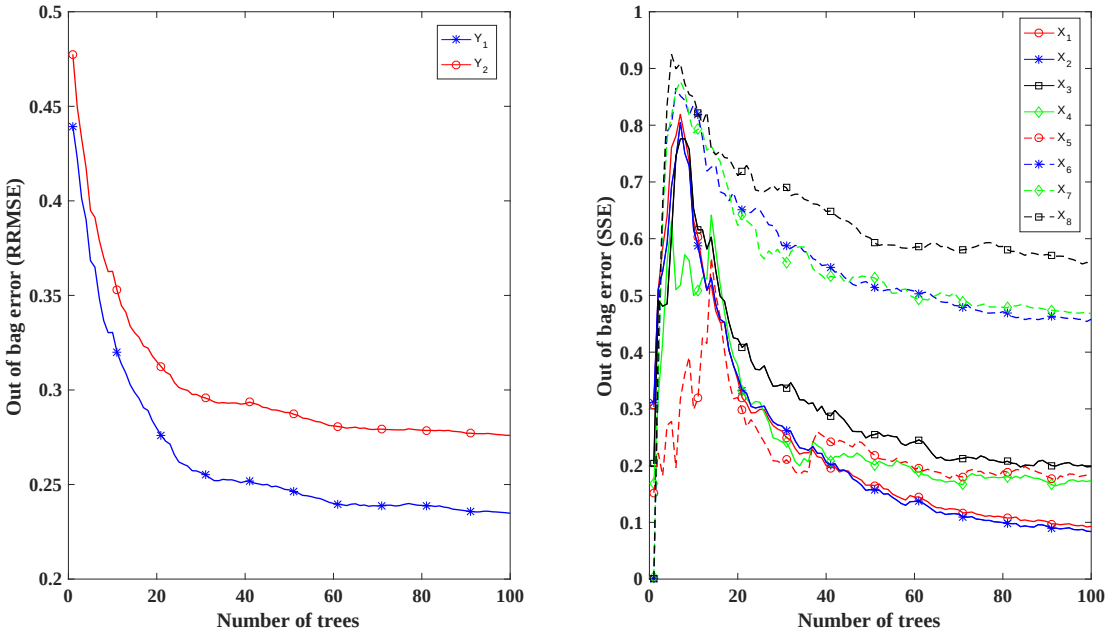

Figure 4.6. Out of bag errors for TRF algorithm using ENB dataset.

4.4.2.2. Robustness Analysis. In order to analyse the robustness of the proposed method in terms of target numbers, we compared TRF method with R-TRF. Figure 4.7 demonstrates a pairwise comparison of RRMSE values for transductive and inductive settings. Transductive experiments refer to predictive performance for the unlabelled data used in training. In inductive experiments totally unseen (test) instances are under consideration. We observe that, almost all error values accumulate around the $y = x$ line. This observation implies consistent results for the data sets with related targets. We can recommend R-TRF for such data sets since it has a lower computational complexity than TRF with a similar performance.

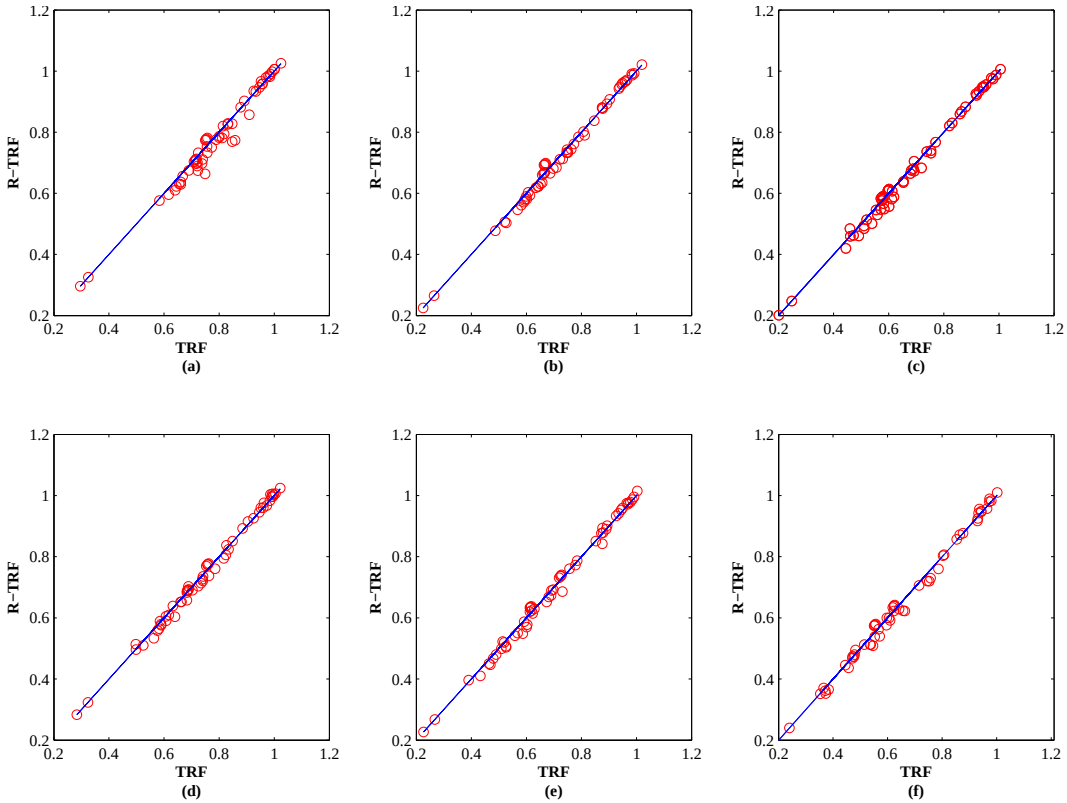

Figure 4.7. RRMSE values of R-TRF and TRF for experiments with (a and d) 5%, (b and e) 10% and (c and f) 20% labelled samples. (Upper and lower panels represent transductive and inductive tests, respectively.)

4.4.2.3. Transductive Evaluation. In transductive experiments test sets consist of all unlabelled instances used during training hence the RRMSE values obtained are not suitable for comparison with respect to labelled proportions. In all Figure 4.8, Figure 4.9 and Figure 4.10 methods with no significant difference are connected with a bar and their positions on the scale represent the ranks of their predictive performances. However, a common observation based on the scales given in Figure 4.8, Figure 4.9 and Figure 4.10 is that, PCT for a single tree is always ranked as the worst and significantly outperformed by other methods. In line with that, in Table A.2 and Table A.2 we observe that all ensemble methods over-perform PCT in terms of predictive performance. Considering those inferences, use of ensemble models can be recommended if data sets permit.

Considering the connected groups and ranks given in Figure 4.8, Figure 4.9 and Figure 4.10, at least any two of ERF, TRF or R-TRF methods perform significantly better than PCTF. Noting that, unsupervised information is exploited as feature variances and/or Gini index in PCTF method. However proposed methods benefit unsupervised information in forms of Euclidean distance and dissimilarity measures, respectively. The results suggest that unsupervised information is captured and smoothness assumption of SSL is modelled better when those forms are used. Figure 4.11 illustrates the pairwise comparison of ERF and TRF algorithms in order to understand the effect of distance and similarity based calculations. Comparing results represented in Figure 4.8, Figure 4.9 and Figure 4.10, and Figure 4.11, particularly in Figure 4.11 (c), we could observe that due to high sensitivity of Nemenyi test, it returns ERF algorithm as the best one, i.e. differences between RRMSE values of the algorithms mostly occur second or third digit after decimal. However by visual inspection of Figure 4.11, we notice that almost all of the RRMSE values scatter around or on the line which implies these algorithms perform almost equivalently.

In addition to RRMSE values given Table A.2, rankings shown on Figure 4.8, Figure 4.9 and Figure 4.10 imply that processing unsupervised information as a whole by means of distances or dissimilarities at instance level lifts the predictive performance of models with individual feature treatment. Multi-task structure of ERF, TRF and R-TRF models helps to exploit task relations. Besides, feature interactions cannot be fully utilized when they are separately involved in training. As a consequence, specifically dissimilarity information appears to be a handy alternative for integrating feature characteristics for a better extent.

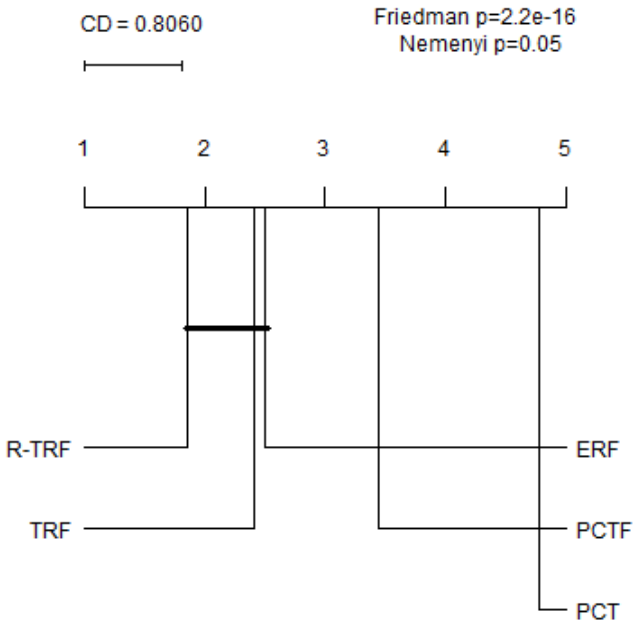

Figure 4.8. Nemenyi test results for transductive setting for data sets given 5% labelled instances.

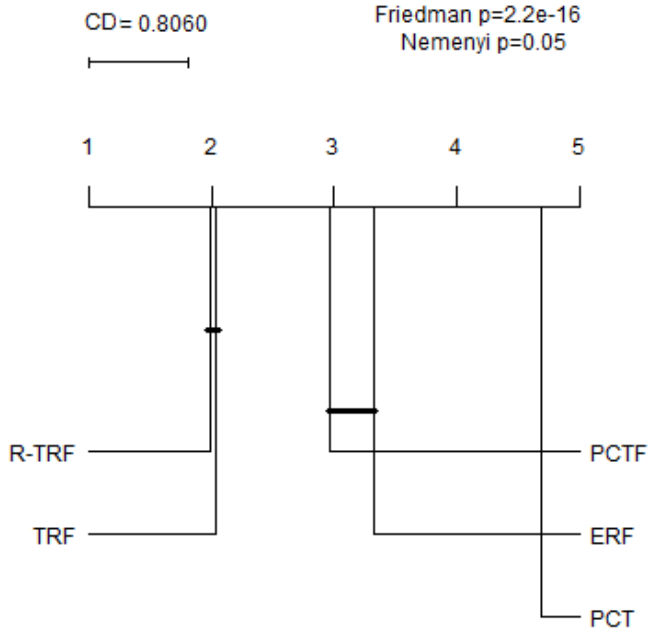

Figure 4.9. Nemenyi test results for transductive setting for data sets given 10% labelled instances.

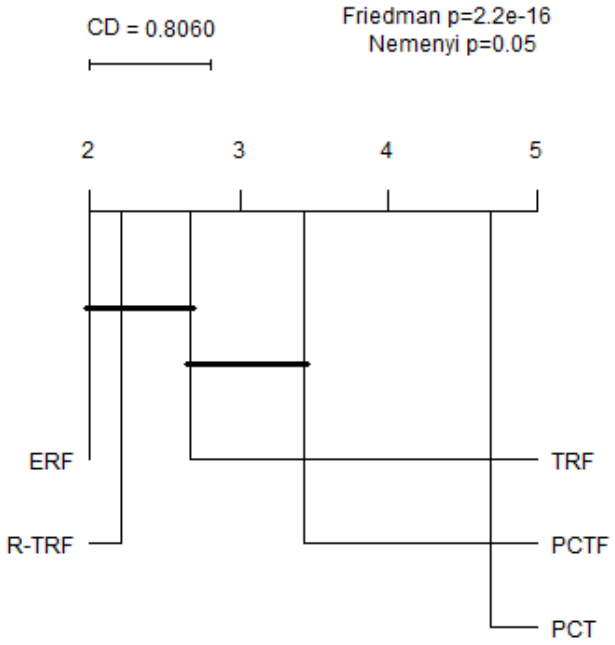

Figure 4.10. Nemenyi test results for transductive setting for data sets given 20% labelled instances.

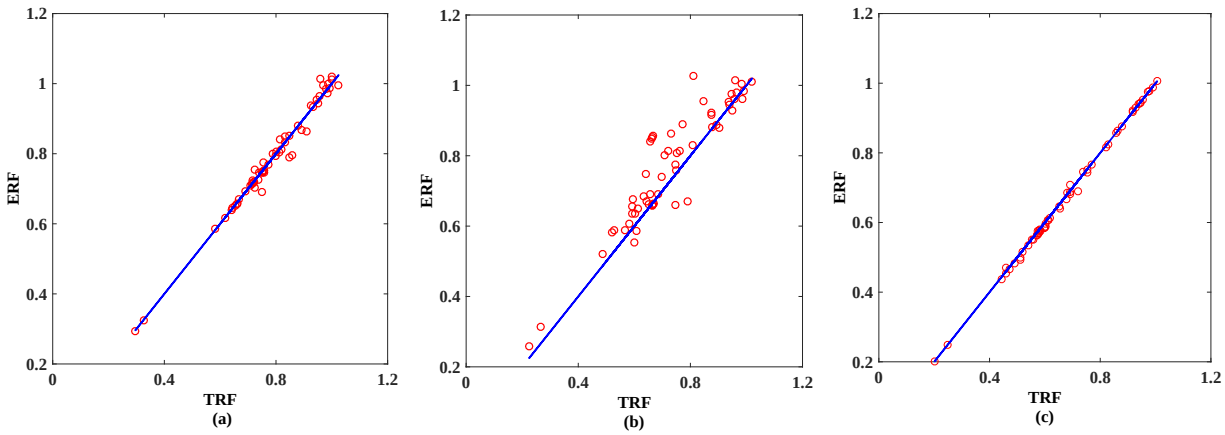

Figure 4.11. RRMSE values of ERF and TRF for transductive setting for experiments with (a) 5%, (b) 10% and (c) 20% labelled samples.

4.4.2.4. Inductive Evaluation. In inductive tests similar to ten-fold cross-validation, we keep one-tenth of the instances as test set and separate them from training process. In line with transductive tests, we compare the algorithm performances by changing proportion of the available labelled part. All three scales given in Figure 4.12, Figure 4.13 and Figure 4.14 represent the ranks of the predictive performances.

Noting that, algorithms without significant difference in terms of RRMSE are combined with a line.

In order to understand whether the differences between performances given in Table A.3 are significant or not, we perform Friedman test. Tests result in extremely small p-values (2.2e-16), hence we cannot reject the significance of differences in performances. Similarly to transductive experiments, PCT is the method with the lowest predictive performance. We can observe from Figure 4.12, Figure 4.13 and Figure 4.14 that TRF is always ranked first and TRF method is not connected to any other methods used except R-TRF. In other words, TRF and R-TRF significantly outperform the other alternatives. In particular, when the number of labelled instances grows, PCTF performances shift towards a better position. Contrary to transductive observations, ERC is outranked by other ensemble methods in inductive setting. This could stem from Euclidean calculations' tendency to overfitting to the unsupervised part which reduces attractiveness of ERC for inductive experiments. In order to be cautious about the high sensitivity of Nemenyi tests, we provide graphical illustrations of RRMSE values for ERF and TRF algorithms in Figure 4.15. All sub-figures (a), (b) and (c) of Figure 4.15 clearly shows the over-performance of TRF compared to ERF as almost all values locate above the line.

Similar to transductive setting, we can observe TRF and R-TRF's success in integrating task relatedness with feature interactions in inductive setting, too. Predictive accuracy of these two methods in experiments with lower number of labelled instances suggests that, TRF and R-TRF models gain higher benefit from semi-supervised learning. This indicates the importance of treating the features in a holistic approach for semi-supervised method design, rather than as separate parts.

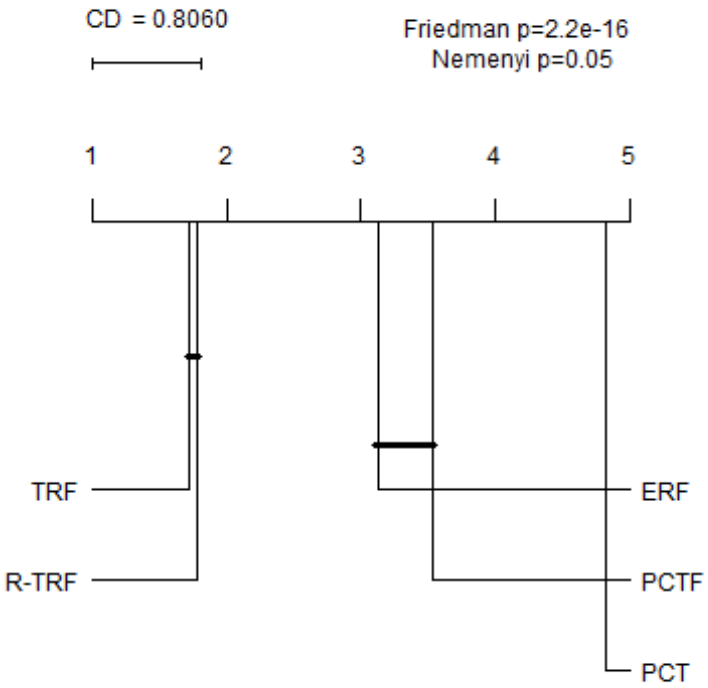

Figure 4.12. Nemenyi test results for inductive setting for data sets given 5% labelled instances.

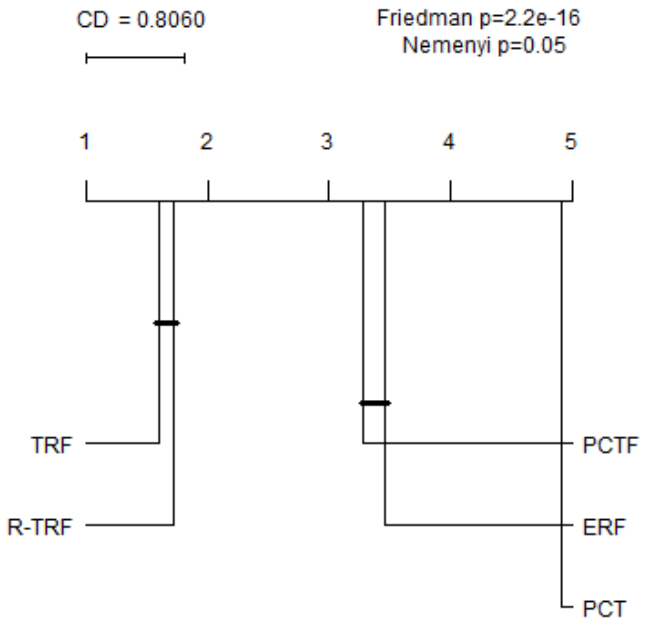

Figure 4.13. Nemenyi test results for inductive setting for data sets given 10% labelled instances.

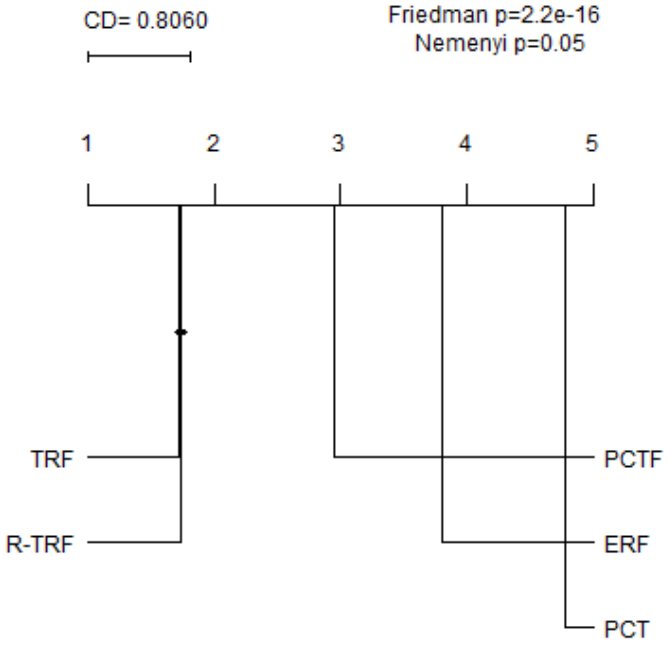

Figure 4.14. Nemenyi test results for inductive setting for data sets given 20% labelled instances.

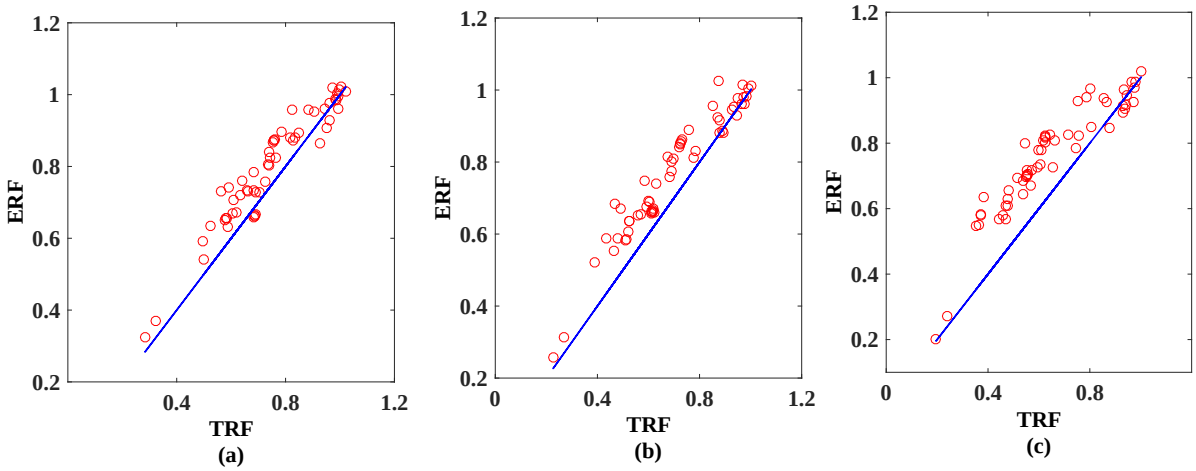

Figure 4.15. RRMSE values of ERF and TRF for inductive setting for experiments with (a) 5%, (b) 10% and (c) 20% labelled samples.

4.4.3. Computational Time Analysis

We design a series of experiments to check whether theoretical complexity of the TRF algorithm agrees with empirical analysis.

We use OES97 data set since the feature, target and instance cardinalities are large enough to observe changes when various conditioning defined over them. Experiments devoted to time analysis were performed under the same technological conditions with previous predictive performance analysis. We randomly select γ proportion of the features, targets and instances. Range of the γ is set as follows: $\gamma \in \{0.2,\ 0.4,\ 0.6,\ 0.8,\ 1\}$. Average of the results obtained by using a TRF ensemble with 100 trees are reported.

Figure 4.16 represents the time elapsed to train a TRF ensemble. On the left hand side of Figure 4.16, we fixed the number of targets to 16 for all experiments. We changed the number of instances (features) by taking a random γ_N (γ_m) proportion of the whole instance (feature) set. We can observe that train time is almost linear with number of features which is consistent with discussions given in Section 4.3.2. Likewise, on the right hand side of the Figure 4.16, we illustrate the required training times where number of features is equal to 263 for all experiments. We can observe a linear increment in training times as the number of targets grows which supports the proposed algorithmic complexity.

Calculations for node splitting procedure constitutes the bottleneck of the computational burden. Assuming the proportion of the features we used in training set to 80%. We observe that, changing the number of training instances results a larger range and considerable differences in required times. However, the number of instances used in training yield more compressed rather than elongated results. Moreover, elapsed times tend to accumulate near to the maximum value unlike the cases given on the left hand side of the Figure 4.16. This characteristic stems from relatively smaller number of split evaluations required values given in the left hand side of Figure 4.16.

Empirical test complexity refers to elapsed time to make a prediction for an instance. This operation requires an instance to traverse 100 trees. Prediction of an instance is nearly instantaneous and after spike the differences between times are almost negligible (see Figure 4.17).

We clearly observe the time advantage of using a global model rather than local one since we would be collecting the predictions one by one from each local learner which would increase the time for prediction. In addition, same advantage of using a global learner is valid for train times.

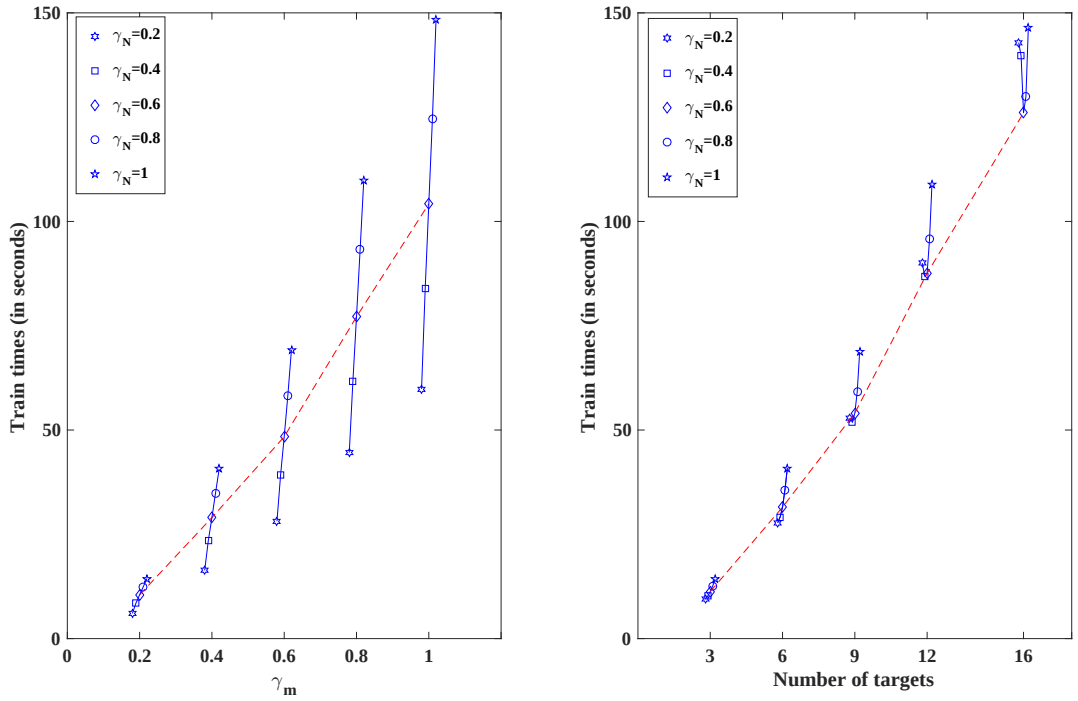

Figure 4.16. Empirical complexity results of training with TRF algorithm for OES97 dataset.

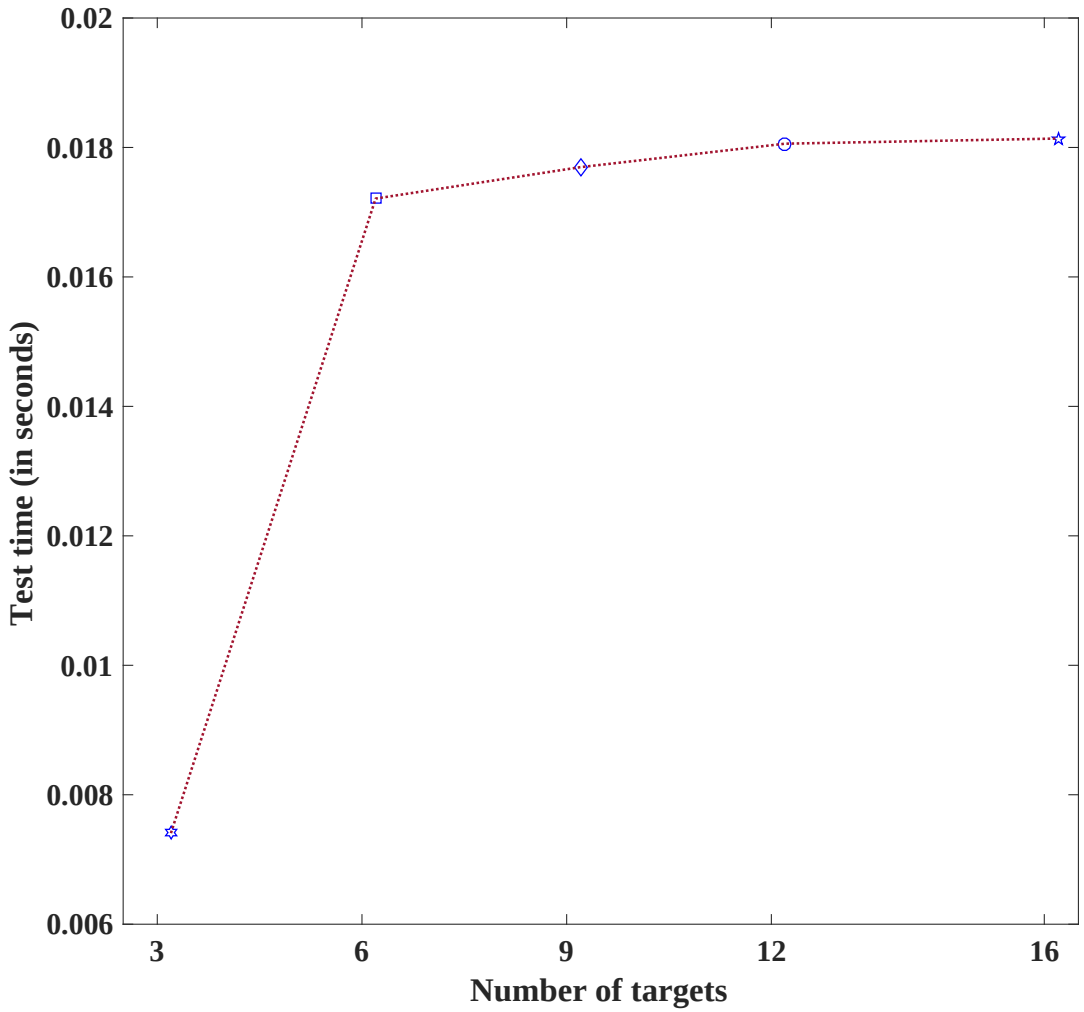

Figure 4.17. Empirical complexity results of testing with TRF algorithm for OES97 dataset.

5. AN ENSEMBLE-BASED SEMI-SUPERVISED FEATURE RANKING FOR MULTI-TARGET PROBLEMS

5.1. Introduction

Handling predictive tasks for multi-target datasets requires addressing some certain properties of those datasets. As it is mentioned and shown in Chapter 3, resolving scale inconsistencies and including target interrelations, have significant roles in success of the classifiers for supervised learning tasks. While dealing with semi-supervised learning applications, considering those properties in building a classifier provides a better learning practice, and therefore a better predictive performance (see Chapter 4). Another challenge that needs to be resolved for multi-target predictive tasks, stems from feature characteristics, i.e. high-dimensionality, irrelevance and redundancy.

Redundant and irrelevant features with high-dimensionality may cause the learning model to fail to deliver the desired results. Therefore, combination of those two issues can be thought as highly problematic. *Feature selection* (FS) or *ranking* (FR) methods help to identify the convenient features and utilize those features in model derivation to enhance the predictive performance of learning model. Moreover, FS methods aid to reduce the dimensionality and as a result they make saving in memory and time. Noting that, numerous studies show the benefits of FS use in both SL and SSL algorithms for single target problems [61] and in SL algorithms for multi-label classification tasks [62], however very few studies can be found for multi-target regression cases [63].

Considering the aforementioned challenges, we focus on to develop an alternative semi-supervised score for FS purpose that does not get adversely affected by scale differences. In Chapter 4, decision trees are extended to a multi-task and semi-supervised form without using weighting schemes or scaling operations. In doing that, we employ a rank based heuristic that selects the candidate with the highest rank as split point.

We set semi-supervised trees as proposed in Chapter 4 as base classifier, we estimate the importance of the selected feature immediately after a splitting step by including information gains from both supervised and unsupervised parts. Once an ensemble is constructed, average of the feature importances are calculated and they are listed in an descending order. After selecting varying proportion of the features, we run the predictive clustering trees [36, 50] with those features and compared with benchmark FR methods [63]. We use predictive clustering trees as base learners as benchmark scores are embedded in their learning process and they present a well-known SSL approach for multi-target learners. Our experimental studies show that the proposed method performs better compared to its state-of-the-art. Specifically, proposed method significantly improves the predictive performance compared to the cases used entire feature set. Besides, proposed FR score outperforms the benchmark scores as the labelled data increase.

The rest of the Chapter 5 is organized as follows: Section 5.2 presents related work and the proposed method is positioned here. Section 5.3 delivers the proposed methodology. Next, in Section 5.4 experimental set-up is given and the results are discussed.

5.2. Related Work

We consider semi-supervised FS and FR methods for multi-target problems hence we limit the related work in respect thereof. The distinction between FS and FR can be expressed via the way they are used in reducing the number of features. The former group of algorithms are used to identify the subset of the most powerful features that improves a learning model's performance yet the latter group's output is an order of the features which considers the relation between the learning tasks and the features [23]. The choice of a taxonomy for dimensionality reduction methods depends upon different concerns.

In [64], those concerns are listed as *i)* learning forms (supervised, unsupervised and semi-supervised), *ii)* connection between learning model and FS method (filter, wrapper and embedded), *iii)* evaluation measures (information theoretic, distance-based, dependency-based), *iv)* search types (forward increment, backward elimination, random search), and *v)* produced results (subset, rank).

We briefly present SSL related FS and FR strategies following the categorization given in [61]. Basically, filter methods require a score measure to evaluate various characteristic of the features. Those characteristics could be based on feature-target relation, i.e. Fisher score and pairwise linkages, or based on properties of the feature in an individual or a relative form, i.e. variance, Laplacian score. To make those scores suitable for SSL, they require proper modifications to include the missing unsupervised or supervised information. Filter methods interact with the learning algorithm in a sequential manner, in other words they return a ranking of the features and a certain amount of them should be selected and those values are used in training.

Wrapper methods are used for in the search of best subset of features in an iterative manner. They require training the learning model multiple times similar to self-training or co-training in SSL approaches. Likewise, recursive elimination of the features is also another strategy used as a wrapper technique and unlike ensemble learning the model is trained until all features are assigned with a rank [65]. Embedded methods return the most useful subset or a ranking of the features as an output of the learning algorithm. In these methods, learning models by-products are exploited such as out-of-bag (OOB) estimations for an ensemble [66] or a regularization parameter in the loss function determines the selected features [67]. Each of those approaches have certain advantages and disadvantages. To summarize, filter methods are known to be fast yet their predictive performance is lower compared to others. Wrapper methods have superior performance yet they are disadvantageous from computational point of view. Since feature selection is not an isolated phase from training process, they have superior accuracy than filter methods. In addition, they do not require recursive training unlike wrapper methods hence their computational burden can be considered as lower.

Majority of FS and FR techniques are designed for handling either numerical or categorical values. In addition to mixed features, those algorithms require weight calculation or transformation operations for targets since supervision of the outputs is required. Noting that, decision trees are able to handle mixed features and various ways to exploit trees for FS and FR are given in the literature [68]. In [27], random forests (RF) are used in estimating variable importances (VI). In calculating VI of a feature, first an RF is created and following that selected feature is permuted for OOB instances. Those instances are pushed down the trees and an overall error is calculated for the selected feature. The higher error outcomes imply the higher importances. In [66], they extend the [27]'s work to semi-supervised setting as VI given therein is valid for supervised application. They use a self-training where the most confident unlabelled instances are included to the labelled examples set. The idea of combining self-training and ensemble learning for FS appears in [69]. Similar to [66]'s study, they enlarge the set of labelled examples with the most confident predictions however they perform FS upon completion of semi-supervised training phase. A natural outcome of self-training is propagation of earlier estimation errors throughout the training. This issue may weaken the outcome either in embedded or wrapper FS.

Another ensemble application of FS with SSL is given in [70] where a boosting mechanism is employed. Increasing the labelled instances depending on the most confident estimations is not limited with self-training approaches, co-training also works in a similar way with a slight difference. Co-training requires separate datasets and classifiers to train on those datasets are used to feed each other. In [71], a co-training scheme is used in identifying the feature relevancies. Despite slight differences in their FS and FR procedures, they are all able to handle single-target problems.

Apart from treating an MTP as a combination of single-target models, few studies on FS and FR applications with multi-target algorithms can be found for classification problems [23, 72, 73]. However, for regression cases the only study uses a multi-target strategy is given in [63] for supervised setting. To the best of our knowledge, our study proposed here presents the first multi-target application of semi-supervised FR for regression problems.

Ranking strategies given in [63] provide the several score variants embedded in predictive clustering trees (PCT). Ensembles of PCTs are able to handle MTP as a weighted sum of target information forms the score function in derivation of PCT [9]. In their study, [63] show the benefit of multi-target strategies in terms of computational performance when they are combined with various scores. Those scores are collected throughout the derivation of the PCTs and are simple heuristics using such as depth or information gain at a given split. In addition to saving time, those algorithms are competitive from predictive power perspective compared to their single-target versions. Besides, the same scores are extended to SSL models for multi-target classification problems in [23]. Their study sets the benchmark models for semi-supervised MTPs.

5.3. Methods

In this section, we refer the preliminaries on derivation of decision trees for supervised and unsupervised tasks as we develop a semi-supervised FR mechanism based on combination of those ideas. Details on use of decision trees for supervised and unsupervised learning are given in Section 2.1.2 and Section 2.1.4. We use the semi-supervised multi-objective trees given in Chapter 4 as base learner. Hence we refer Section 4.3 for details on derivation of semi-supervised trees.

5.3.1. Feature Ranking

We propose a multi-objective score that captures both supervised and unsupervised characteristics of the features (see Figure 5.1). Separate calculation of feature contributions in learning processes via Gini index or other type of indicators, is a trivial option. However, as it is shown in Chapter 4, semi-supervised applications are better in reflecting the necessary data characteristics. Hence, we aim to provide a unified form to capture the feature contributions that considers potential semi-supervised contribution throughout the training phase.

In evaluating performance of a solution consists of several criteria, various indicators are proposed in multi-objective optimization domain [74].

In that domain, usually a set of optimal solutions exist and is characterized as *Pareto front* which consists of solutions that are not dominated by any other feasible decision vector. A specific category of aforementioned indicators are based on the convergence to Pareto frontier and in doing that, usually reference points are used. Following that, we build a scoring strategy based on evaluation of the performance a feature similar to the approximation to Pareto frontier by using *ideal point*. An ideal point refers to the minimum solution of each single criterion and in our calculations we fixed it to origin. At each split step we calculate a semi-supervised score (SSS) subsequent to split decision.

Require Training set D with n instances, stopping conditions
if termination criterion **then**
  Create leaf node and label it with its target averages;
  Return leaf node;
else
  Draw a P^* random subset of the input variables;
  Examine all possible binary splits for each of the input subset P^*;
  Create the set of first frontier with respect to the quality measure in vector form;
  Select the split with minimum average rank;
  Calculate SSS of the selected feature for split;
  Apply the best split and create left node and right node;
  Return Recursive Partitioning(left node);
  Return Recursive Partitioning(right node);
end if

Figure 5.1. Rank Based Semi-Supervised Tree and SSS Algorithm.

We illustrate SSS calculation with a toy example given in Figure 5.2. We assume that in Figure 5.2 c_1 and c_2 axes represent two criteria to be assessed.

In this case, we set c_1 and c_2 are supervised (e.g. SSE of a target) and unsupervised (e.g. overall distance from node medoid) scores obtained from a partition based on feature f. Let d_0, d_1 and d_2 be Euclidean distances of a parent node and its child nodes considering (c_1^i, c_2^i) coordinates from origin. SSS of feature f is the result of the relation given in Equation (5.1). In Equation (5.1), n_i values refer to number of instances placed in node j.

$$SSS_f = d_0 - (\frac{n_1}{n_0}d_1 + \frac{n_2}{n_0}d_2) \tag{5.1}$$

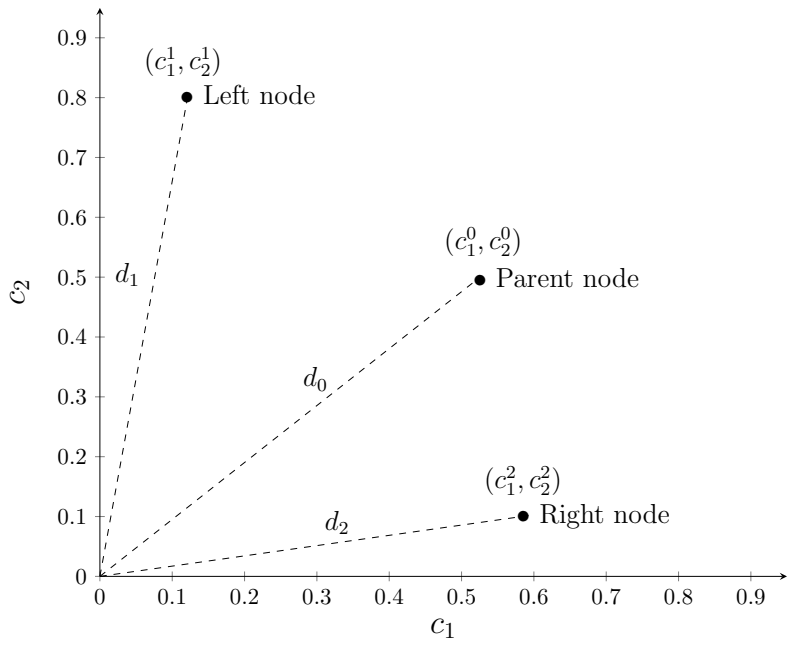

Figure 5.2. Toy example for semi-supervised feature score.

In our calculations supervised c_i scores consist of target-wise SSE values for labelled instances. Unsupervised c_i scores are the values of within cluster distance from node medoids divided by root within cluster distance from root medoid. To clarify, for a dataset with m number of targets, elements of c_i vector is equal to $m + 1$, hence associated d_i distance is calculated based on those dimensions.

As a semi-supervised tree in the ensemble grows, an SSS value is recorded for the most beneficial feature. By scanning entire ensemble, SSS values for each feature are combined and normalized by the number of trees. At the end of that procedure features are related with an SSS (see Figure 5.1). Ranks of the features are obtained by listing SSS values in descending order.

5.3.2. Algorithmic Complexity

A standard decision has a computational complexity of $\mathcal{O}(PN \log N)$ where P and N refer to number of features and instances, respectively. Since for dissimilarity calculations a TRT ensemble with size J is used, an additional computational complexity of $\mathcal{O}(JN \log N)$ occurs. In semi-supervised trees $\sqrt{P}$ features are used in split evaluation. All targets are considered one by one, hence complexity of computations at each split becomes multiple of number of targets, that is M. Besides, we also need to consider an additional U number of unsupervised criteria at each split, overall computational complexity of a single semi-supervised tree $\mathcal{O}((M+U)\sqrt{P}N \log N)$. For an ensemble with size J, complexity of a semi-supervised tree is concluded as $\mathcal{O}(J(M+U)\sqrt{P}N \log N)$. Besides, for SSS calculations performed at each split, an additional $\mathcal{O}(JN \log N)$ operations required.

5.3.3. Benchmark Scores and Predictive Model

Scores given in [63]'s are the state-of-the art measures for MTP. Besides, for semi-supervised MTP [23] employ those scores for multi-target classification problems. In our study, we use *Symbolic, Genie3*, and *random forest importance* (RFI) scores for comparison purpose. Noting that all benchmark scores are produced by using PCTs proposed by [9]. Symbolic score collects the counting information about the features used in node partitions as the more important a feature is, the more it appears in the partitions. Impurity reduction information at a given split forms the Genie3 scores for the features. Similar to Symbolic score, information for Genie score of a feature is the outcome of a split decision. Unlike Symbolic and Genie3 scores, RFI value for a feature requires assessing OOB errors for original and permuted versions of a feature.

Greater predictive performance reduction for OOB samples implies higher degree of importance for a feature.

5.4. Experiments and Results

Datasets used in the experiments are summarized in Table 5.1 and are available in the project website given in [6]. We compare the predictive performances by using PCTs as they provide a learning scheme based on multi-target architecture. A WEKA based Java implementation of the PCT is publicly available as CLUS software and is shared in [63]. We obtain Symbolic, Genie3 and RFI scores by using CLUS for all the datasets. Proposed SSS ranks are obtained by a routine written in MATLAB R2016a. Besides, in calculation of similarities we use an implementation of totally random trees in Python 3.6.5 environment [60]. By using the same parametrizations given in [63], we collect the benchmark scores with 100 bagged trees. Besides, we create a multi-objective ensemble with the same size for SSS ranks. We perform all experiments with 10-fold cross-validation via a notebook with MS operating system and 16 GB RAM. With available four physical cores (i7-7700HQ, 2.8 GHz) we use single thread for all experiments. We set the same train and test instances in the experiments for fair comparison. Implemented codes can be found within the github project [41]. We run the experiments with various proportions for labelled instances. We randomly select 5%, 10% and 20% of the data and keep those instances labelled.

Table 5.1. Datasets for semi-supervised feature ranking experiments.

Dataset	**Samples** (N)	**Features** (p)	**Targets** (m)
Water Quality	1060	16	14
OES10	403	298	16
OES97	334	263	16
ATP1D	337	411	6
ATP7D	296	411	6
MP5	700	80	4
MP6	700	80	4

In evaluating predictive performances we collect the results in transductive setting. To clarify, we consider entire unlabelled parts of the datasets that are used in training phase. We compare predictive performance of full set of features case against FR applied datasets. Once an FR method returns a ranking result, 50% of the features are selected from the ordered list of ranks. We perform statistical analyses based on the outcomes of original and reduced datasets. Assessments are based on relative root mean squared error (RRMSE), we refer Equation (3.1) for RRMSE formula.

5.4.1. Results and Discussions

We compare the effect of various FR strategies on predictive performances by using the statistical methods as suggested in [42]. Initially we perform Friedman test as a non-parametric alternative of ANOVA to observe whether the difference in performances is statistically significant or not. Besides, Nemenyi test is used to analyse the significance of precedence in terms of predictive performance. Average rank results of the Nemenyi test are ordered on a scale where the candidates with no statistical difference are connected with a bar. All RRMSE results are given in Table A.4.

In Figure 5.3, Figure 5.4 and Figure 5.5 results for original feature sets are represented with No FR acronym. Considering Figure 5.3, Figure 5.4 and Figure 5.5, benefit of applying a FR step is clearly seen as No FR alternative takes a position at the right most of the scales. In other words, full feature set yields the last or second to the last result in terms of predictive performance. Figure 5.3 shows that, Symbolic score performs well for the cases with limited label information (e.g. 5%). However, as the labelled instance proportion increases, SSS surpasses the rest and yields the best rank.

Both in Figure 5.4 and Figure 5.5 Symbolic score takes the second place following SSS. Considering the overall performance of SSS and Symbolic scores, we can deduce that those two methods are able to capture feature characteristics in terms of semi-supervised contribution.

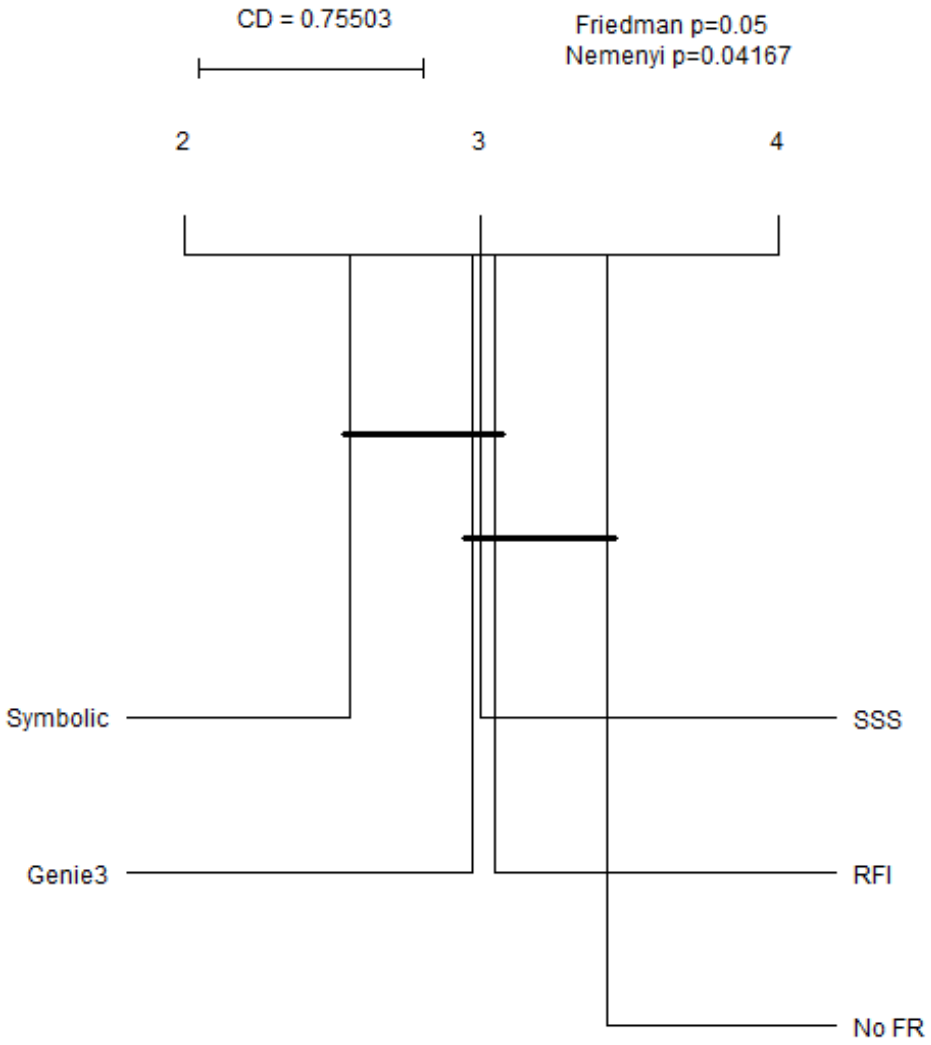

Figure 5.3. Average rank diagrams for experiments with 5% labelled instances.

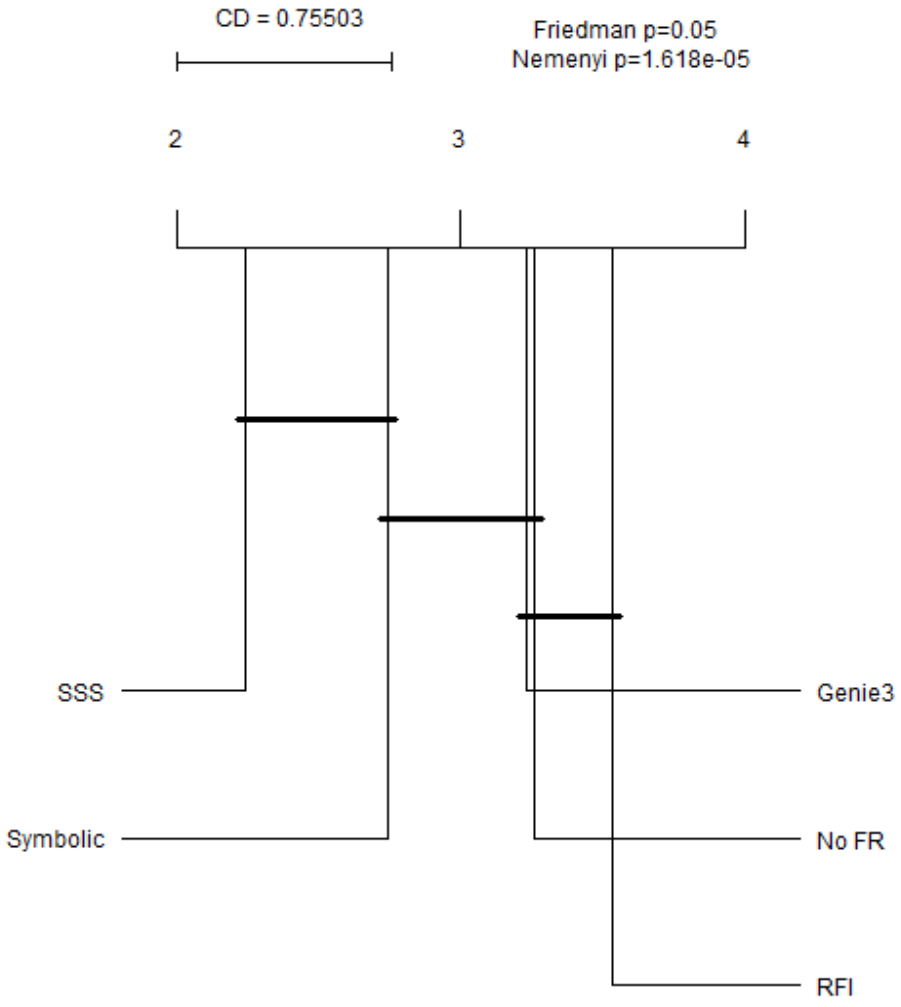

Figure 5.4. Average rank diagrams for experiments with 10% labelled instances.

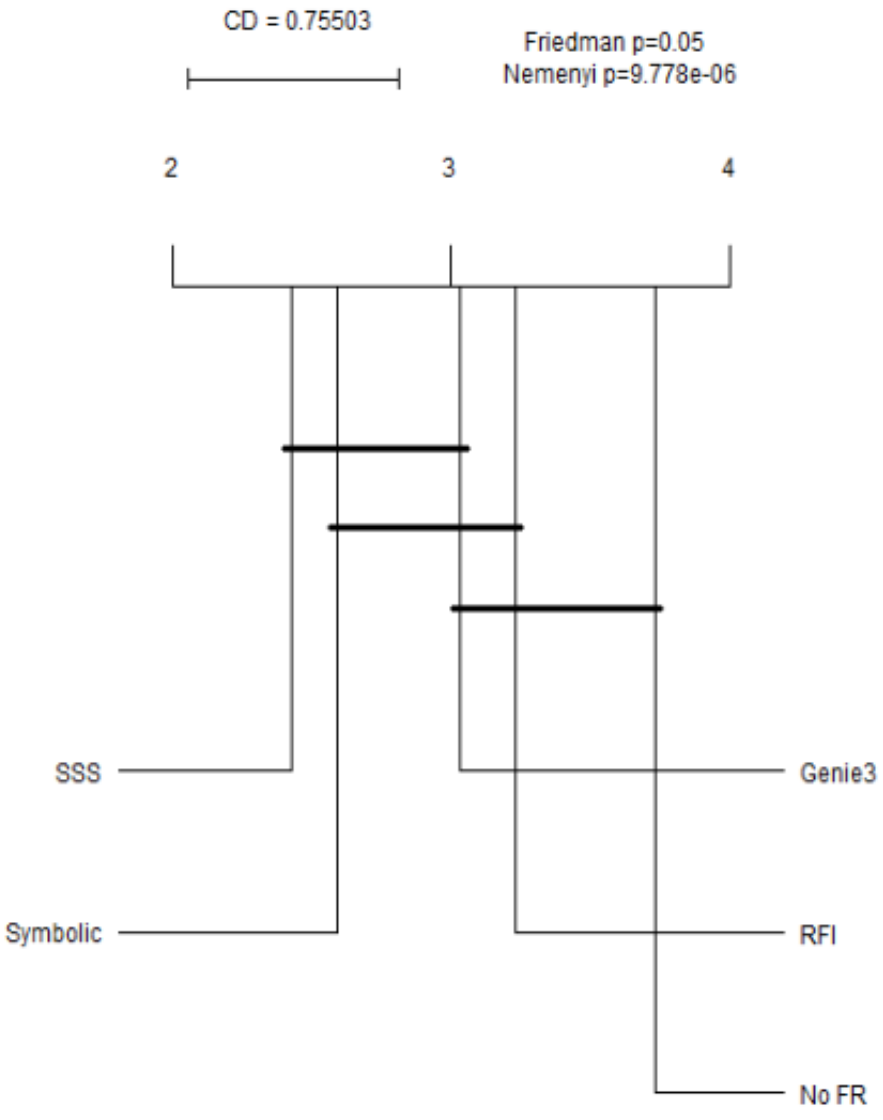

Figure 5.5. Average rank diagrams for experiments with 20% labelled instances.

It is worth noting that, RFI is outperformed by its benchmark scores and difference of average ranks for RFI and No FR is not statistically significant. For 10% case, RFI yields even a worse average rank compared to No FR case. Considering average rank placements of the methods on diagrams, we observe that differences between performances can be seen more clear when the labelled information increases. To clarify, for 5% (e.g. Figure 5.3) and 10% (e.g. Figure 5.4) labelled cases performances of several methods are not distinguishable as clear as given in 20% (e.g. Figure 5.5). We observe that semi-supervised FR improves the overall performance of PCTs in predictive tasks.

Since Figure 5.3, Figure 5.4 and Figure 5.5 demonstrate that an increment in labelled data provides better results, we decompose the supervised and unsupervised contributions in calculation of SSS. To clarify, only supervised (or unsupervised) criteria is used for SSS calculation. In Figure 5.6 criteria that construct SSS are given for two datasets, i.e. OES97 and MP5. We select those datasets due to the difference in their number of targets as that property may aid to understand the effect in a more clear way. Horizontal axis for diagrams given Figure 5.6 consist of ranked features. To clarify, first and last terms in axis are the best and worst ranked features for different experimental settings. Considering the curves given Figure 5.6, trajectories are steeper in approximately half of the features compared to the other part. That observation may support the representative power of the selected features. Wide distances between supervised and unsupervised curves in the upper panel of Figure 5.6 show that supervised contribution is higher than unsupervised part. In the lower panel of Figure 5.6, a similar pattern with the upper panel can be seen, however distances between trajectories become shorter. This observation is expected since OES97 dataset contains a greater number of targets compared to MP5 dataset.

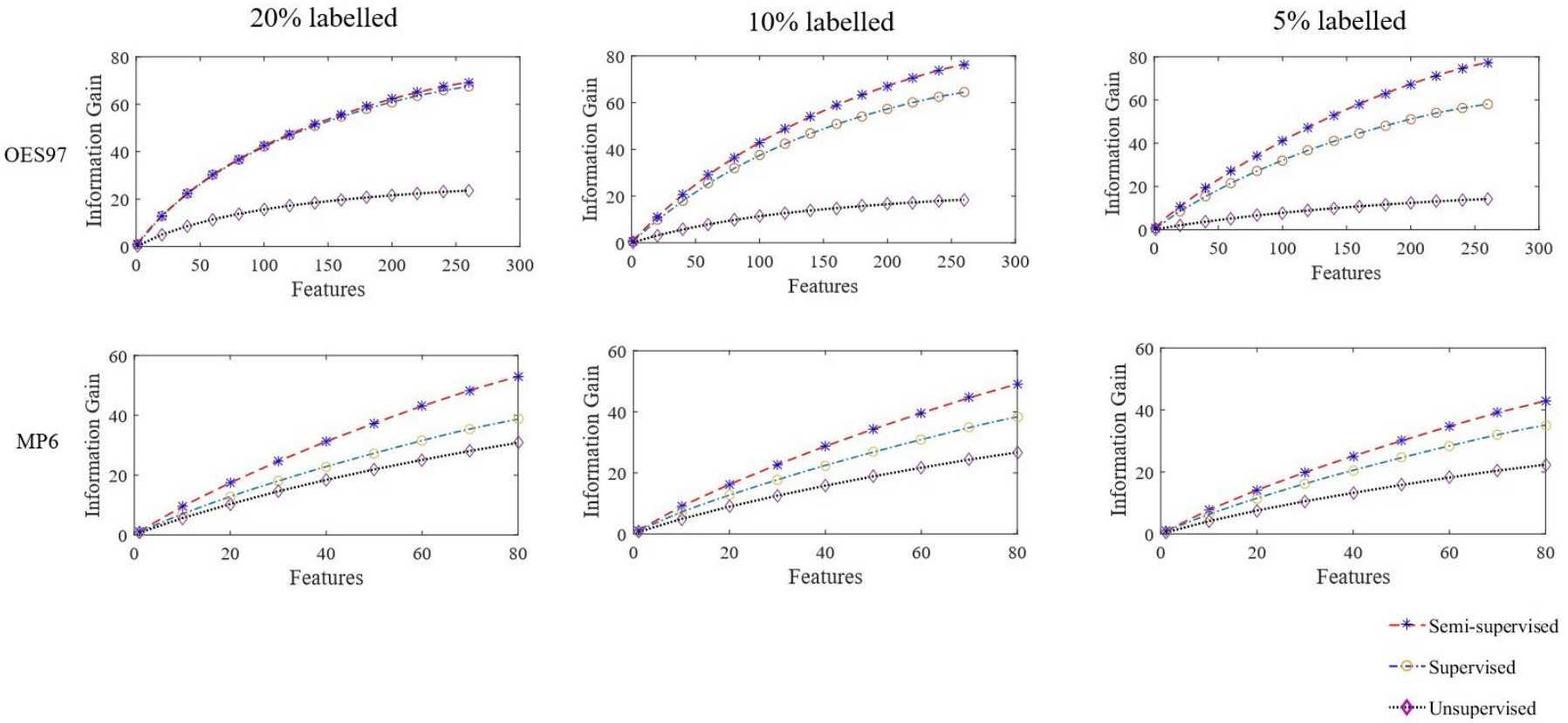

Figure 5.6. Overall information gains for OES97 and MP6 datasets.

6. CONCLUSIONS AND FUTURE WORK

6.1. Conclusions

In this thesis, we investigate multi-objective approaches for multi-target learning. While doing that, we aim to address scale inconsistencies observed in the multi-target datasets. Another concern we aim to solve is utilization of target interrelations. Towards those aims, we propose multi-objective decision trees besides an alternatives chaining policy for supervised problems. We extend multi-objective trees to their semi-supervised version so that proposed models are able to show competitive predictive performance. In addition, we propose a semi-supervised feature ranking scheme as a subsequent work of semi-supervised learning methods given here.

In the first study, we investigate and compare two streams of target exploitation in learning processes for multi-target datasets. We aim to handle multi-target prediction problems with scale differences in their targets and utilize target relations in training simultaneously. We use decision trees as base learners and propose a multitask extension, and a data-driven space expansion strategy. Our experiments show that, utilizing targets as additional inputs with proper sequence improves single target approach. Especially, SC method shows superior performance datasets with larger number of targets. In addition, proposed multitask extension makes better predictions than its multi objective counterpart. Both of these techniques go through the difficulties arises scaling of data. Besides, they are able to transfer relevant information to the training phase so that it can further improve the generalization power of the classifier.

In the second study, we consider semi-supervised learning problem for multi-output data sets. Usually multi-output learning problems are handled by local learners however learning each target in isolation results missing target relations. It becomes more problematic in semi-supervised learning applications since the accessible label information is limited and a valuable potential of information would be overlooked.

In addition, target scales usually do not agree hence data transformation is inevitable for classifiers to work properly. However, transformation operations are criticized due to several reasons such as loss or distortion of relational properties of the data. To solve those issues, we propose to extend standard decision trees to multi-task learners by introducing a score measure in a multi-criteria form that is used in split evaluations.

Scale inconsistency issue is not only observed in targets, it is actually more common in features. Decision trees are inherently robust to features with scale differences for supervised learning applications, however this property may disappear in semi-supervised setting. To clarify, semi-supervised applications require unsupervised indicators which are defined over features. Noting that, defining an unsupervised indicator that properly reflects the data characteristics is a challenging task. Moreover, with the increase in number of features, capturing unsupervised information becomes harder due to the curse of dimensionality. To exemplify, classical distance measures are widely used as unsupervised indicators but they may result misleading inferences when data set is subject to given condition. In order to cope with aforementioned issue, we argue that dissimilarity information performs better in preserving and discovering the original properties of the data. We use total random forests to derive the similarities between instances. By converting them into dissimilarities, we model and combine the unsupervised information with supervised criteria and obtain semi-supervised score measure.

Experimental studies show that proposed algorithm outperforms its benchmark that is semi-supervised predictive clustering forests. Besides its predictive performance, our approach captures feature interrelations better than its benchmark as we evaluate instance similarities as a whole. In addition, since our approach is a global one, it has a reduced computational complexity than local learners without sacrificing task relatedness.

In the third study, we aim to handle FR for multi-target datasets with limited label information.

Generally, for each target a separate classifier is trained yet that strategy is criticised for failing to include target interrelations a consequence of being local learners. Another issue with multi-target datasets arises when targets have scale inconsistencies. In addition to local learning, that challenge is resolved by using various transformation methods. However, those operations may change the statistical properties of the data. Besides, redundant or irrelevant features further complicate the learning process and as a result they may degrade the predictive performance. Considering aforementioned issues, an alternative that handles scaling issues without losing target interrelations is the main motivation of this study. More specifically, we focus on selecting the most promising features and enhancing semi-supervised learning techniques by building learners based on those selected features.

We extend multi-objective trees to make them suitable for capturing semi-supervised characteristics of the data. In derivation of those trees we integrate an FR procedure to collect the features' contribution during learning process. We observe that, proposed technique is ranked first and outperforms its state-of-the-art benchmark FR scores as the labelled data increase.

We can summarize the contributions of this thesis work as follows: We propose a multi-objective tree structure that considers the contribution of target that belongs to a multi-target dataset. Besides, we provide a data-driven chaining strategy based on prioritization of the targets with the largest contribution to the learning model. There two aims of those proposed strategies. First we focus on handling the scale inconsistencies observed in targets. Second, we aim to use the potential contribution of target relations.

Another contribution of this thesis work can be given as providing a tree based semi-supervised learning model that aims to deal with handling mixed type of targets and features. We extend the proposed supervised model to its semi-supervised version by including unsupervised criteria obtained via totally randomized trees. By doing so, we aim to take advantage of unsupervised use of random forests in order to address the required distance based calculations for features.

In addition, we propose a semi- supervised feature ranking procedure that is embedded to tree derivations. Proposed score is inspired by quality measures used in multi-objective optimization domain and unifies the features' contribution to the learning process by considering both target and feature perspectives.

6.2. Future Work

6.2.1. Target Cardinality

From practical point of view, SC method can be used for discovering potential dependencies between targets. Alternatively, SC can be used for testing the confidence of an initial knowledge on target interrelations. Proposed multi-objective designs or ERC methods cannot offer this opportunity. Apart from that, dealing with optimization problems with more than three objectives, are known as many-objective problems and evaluating trade-offs for such kind of problems becomes harder as the number of objectives increases [75]. For many-objective problems in multi-objective optimization domain, that research line can also be useful in reducing or handling many-objective cases.

Despite its predictive performance, a possible pitfall of our classifier chain design may arise when target cardinality is extremely large. This is due to number of evaluations required to discover a good chain. Developing alternative tools for competitive chaining policies with reduced computational costs could be a future direction.

6.2.2. Interactive Learning Extensions

Semi-supervised learning applications deal with learning cases with limited label information. As it is stated before, manually labelling and increasing the available information is an option. However, this is usually not preferred as manually labeling may require expert opinion, expensive equipment or experiments with long duration. Contrary to semi-supervised learning, interactive learning applications try to make the best use of manually labelling opportunities given a limited budget for querying.

Here, the aim is to select the most informative queries to perform so that the learning trajectory improves at its highest. In a sense, those two learning approaches try to work well under similar conditions, i.e. limited label information. Here, if those approaches are integrated, semi-supervised learning would aid interactive mechanism to make easier decisions on refining the candidate queries. To clarify, instances which a semi-supervised classifier returns high confident results, can be eliminated from query set. Integration of those approaches can be given as an alternative future direction of this thesis study.

6.2.3. Computational Improvements

As we perform our experiments in an ensemble setting, one future application can be based on parallelization of the methods for the sake of better time results when dealing with large datasets. Since each tree in the random forests could be independently trained, parallel training of the trees is a strategy that is commonly in use. Besides, training an independent model for each target is possible in chaining, therefore a strategy that is based on target-wise decomposition could be a different alternative for parallelization. Another direction for future research can be on implementation of SC algorithms for other kind of classifiers as it is ready to extend for usage. Chaining strategy is readily available for adaptations with alternative base classifiers since the method requires separate training sessions for each of the targets. Noting that, in SC strategy out-of-bag (OOB) errors are used for chaining decisions. A major merit of OOB error use is that, it prevents the learning model from overfitting. Therefore, if the selected base classifier does not allow to calculate OOB errors unlike random forests, an internal cross validation (CV) mechanism should be employed to measure the performance of the trained models. Noting that, internal CV calculations result additional computational burden in model derivation, hence this strategy could be considered as more time-consuming compared to OOB calculations.

Despite its simplicity in use, a possible limitation of the proposed semi-supervised approaches for both predicitve and FR tasks, arises when the sample size gets very large.

Here, despite the merit of dissimilarity information use in the proposed method, required calculations could be problematic when the data gets larger. Hence, efficient ways on discovering feature interrelation information can be given as another possible search of direction.

REFERENCES

1. Schweidtmann, A. M., A. D. Clayton, N. Holmes, E. Bradford, R. A. Bourne and A. A. Lapkin, "Machine Learning Meets Continuous Flow Chemistry: Automated Optimization Towards the Pareto Front of Multiple Objectives", *Chemical Engineering Journal*, Vol. 352, pp. 277 – 282, 2018.

2. Criminisi, A., J. Shotton and E. Konukoglu, "Decision forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-supervised Learning", *Foundations and Trends in Computer Graphics and Vision*, Vol. 7, No. 2-3, pp. 81–227, 2012.

3. Hu, Y., Z. Bie, T. Ding and Y. Lin, "An NSGA-II based Multi-objective Optimization for Combined Gas and Electricity Network Expansion Planning", *Applied Energy*, Vol. 167, pp. 280 – 293, 2016.

4. Waegeman, W., K. Dembczyński and E. Hüllermeier, "Multi-target Prediction: A Unifying View on Problems and Methods", *Data Mining Knowledge Discovery*, Vol. 33, No. 2, pp. 293–324, 2019.

5. Spyromitros-Xioufis, E., G. Tsoumakas, W. Groves and I. Vlahavas, "Multi-target Regression via Input Space Expansion: Treating Targets as Inputs", *Machine Learning*, Vol. 104, No. 1, pp. 55–98, 2016.

6. Melki, G., A. Cano, V. Kecman and S. Ventura, "Multi-target Support Vector Regression via Correlation Regressor Chains", *Information Sciences*, Vol. 415, pp. 53–69, 2017.

7. Read, J., L. Martino, P. M. Olmos and D. Luengo, "Scalable Multi-output Label Prediction: From Classifier Chains to Classifier Trellises", *Pattern Recognition*, Vol. 48, No. 6, pp. 2096 – 2109, 2015.

8. Dembczyński, K., W. Waegeman, W. Cheng and E. Hüllermeier, "On Label Dependence and Loss Minimization in Multi-label Classification", *Machine Learning*, Vol. 88, No. 1, pp. 5–45, Jul 2012.

9. Kocev, D., C. Vens, J. Struyf and S. Džeroski, "Tree Ensembles for Predicting Structured Outputs", *Pattern Recognition*, Vol. 46, No. 3, pp. 817–833, 2013.

10. Lallemand, J., O. Pauly, L. Schwarz, D. Tan and S. Ilic, "Multi-task Forest for Human Pose Estimation in Depth Images", *International Conference on 3D Vision - 3DV 2013*, pp. 271–278, June 2013.

11. Gao, Y., Y. Shao, J. Lian, A. Z. Wang, R. C. Chen and D. Shen, "Accurate Segmentation of CT Male Pelvic Organs via Regression-based Deformable Models and Multi-task Random Forests", *IEEE Transactions on Medical Imaging*, Vol. 35, No. 6, pp. 1532–1543, 2016.

12. Lin, S., B. Azarnoush and G. C. Runger, "CRAFTER: A Tree-Ensemble Clustering Algorithm for Static Datasets with Mixed Attributes and High Dimensionality", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 30, No. 9, pp. 1686–1696, 2018.

13. Van Engelen, J. E. and H. H. Hoos, "A Survey on Semi-supervised Learning", *Machine Learning*, Vol. 109, No. 2, pp. 373–440, 2020.

14. Pise, N. N. and P. Kulkarni, "A Survey of Semi-Supervised Learning Methods", *In Proceedings of International Conference on Computational Intelligence and Security*, pp. 30–34, 2008.

15. Schwenker, F. and E. Trentin, "Pattern Classification and Clustering: A Review of Partially Supervised Learning Approaches", *Pattern Recognition Letters*, Vol. 37, p. 4–14, 2014.

16. Levatic, J., D. Kocev, M. Ceci and S. Deroski, "Semi-supervised Trees for Multi-

target Regression", *Information Sciences*, Vol. 450, No. C, p. 109–127, 2018.

17. Levatic, J., M. Ceci, D. Kocev and S. Dzeroski, "Semi-supervised Classification Trees", *Journal of Intelligent Information Systems*, Vol. 49, pp. 461–486, 2017.

18. Skolidis, G. and G. Sanguinetti, "Semisupervised Multitask Learning with Gaussian Processes", *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 24, No. 12, pp. 2101–2112, 2013.

19. Du, X., "Semi-supervised Learning of Local Structured Output Predictors", *Neurocomputing*, Vol. 220, pp. 151 – 159, 2017.

20. Navaratnam, R., A. W. Fitzgibbon and R. Cipolla, "The Joint Manifold Model for Semi-supervised Multi-valued Regression", *IEEE 11th International Conference on Computer Vision*, pp. 1–8, 2007.

21. Sheikhpour, R., M. A. Sarram, S. Gharaghani and M. A. Z. Chahooki, "A Survey on Semi-supervised Feature Selection Methods", *Pattern Recognition*, Vol. 64, pp. 141–158, 2017.

22. Li, Y., T. Li and H. Liu, "Recent Advances in Feature Selection and Its Applications", *Knowledge and Information Systems*, Vol. 53, pp. 551–557, 2017.

23. Petković, M., S. Džeroski and D. Kocev, "Ensemble-Based Feature Ranking for Semi-supervised Classification", *International Conference on Discovery Science*, pp. 290–305, Springer, 2019.

24. James, G., D. Witten, T. Hastie and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R*, Springer Publishing Company, 2014.

25. Jin, Y. and B. Sendhoff, "Pareto-based Multiobjective Machine Learning: An Overview and Case Studies", *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, Vol. 38, No. 3, pp. 397–415, 2008.

26. Ehrgott, M., *Multicriteria Optimization*, Springer-Verlag, Berlin, Heidelberg, 2005.

27. Breiman, L., "Random Forests", *Machine Learning*, Vol. 45, No. 1, pp. 5–32, 2001.

28. Gan, G., C. Ma and J. Wu, *Data Clustering: Theory, Algorithms, and Applications (ASA-SIAM Series on Statistics and Applied Probability)*, Society for Industrial and Applied Mathematics, USA, 2007.

29. Geurts, P., D. Ernst and L. Wehenkel, "Extremely Random Trees", *Machine Learning*, Vol. 63, No. 1, pp. 3–42, 2006.

30. Chapelle, O., B. Schölkopf and A. Zien, *Semi-Supervised Learning*, The MIT Press, Cambridge, Massachusetts, 2006.

31. Pan, S. J. and Q. Yang, "A Survey on Transfer Learning", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 22, No. 10, pp. 1345–1359, 2010.

32. Caruana, R., "Multitask Learning", *Machine Learning*, Vol. 28, pp. 41–75, 1997.

33. Zhang, Y. and Q. Yang, "A Survey on Multi-task Learning", *arXiv preprint arXiv:1707.08114*, 2017.

34. Jonschkowski, R., S. Höfer and O. Brock, "Patterns for Learning with Side Information", *arXiv preprint arXiv:1511.06429*, 2015.

35. Glocker, B., O. Pauly, E. Konukoglu and A. Criminisi, "Joint Classification-Regression Forests for Spatially Structured Multi-object Segmentation", A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato and C. Schmid (Editors), *Computer Vision – ECCV 2012*, pp. 870–881, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

36. Kocev, D., C. Vens, J. Struyf and S. Džeroski, "Ensembles of Multi-Objective Decision Trees", J. N. Kok, J. Koronacki, R. L. d. Mantaras, S. Matwin, D. Mladenič and A. Skowron (Editors), *Machine Learning: ECML 2007*, pp. 624–631, Springer

Berlin Heidelberg, Berlin, Heidelberg, 2007.

37. Czajkowski, M. and M. Kretowski, "A Multi-objective Evolutionary Approach to Pareto Optimal Model Trees. A Preliminary Study", C. Martín-Vide, T. Mizuki and M. A. Vega-Rodríguez (Editors), *Theory and Practice of Natural Computing*, pp. 85–96, Springer International Publishing, 2016.

38. Van den Berg, R. A., H. C. Hoefsloot, J. A. Westerhuis, A. K. Smilde and M. J. van der Werf, "Centering, Scaling, and Transformations: Improving the Biological Information Content of Metabolomics Data", *BMC Genomics*, Vol. 7, No. 142, pp. 1–15, 2006.

39. Changyong, F., W. Hongyue, N. Lu and X. M. Tu, "Log Transformation: Application and Interpretation in Biomedical Research", *Statistics in Medicine*, Vol. 32, No. 2, pp. 230–239, 2016.

40. Deb, K., A. Pratap, S. Agarwal and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II", *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, pp. 182–197, April 2002.

41. Adıyeke, E., "Codes and indices", `http://github.com/eadiyeke`, 2020, accessed: August 2020.

42. Demšar, J., "Statistical Comparisons of Classifiers over Multiple Data Sets", *Journal of Machine Learning Research*, Vol. 7, pp. 1–30, 2006.

43. Zuluaga, M., G. Sergent, A. Krause and M. Püschel, "Active Learning for Multi-Objective Optimization", S. Dasgupta and D. McAllester (Editors), *Proceedings of the 30th International Conference on Machine Learning*, Vol. 28, pp. 462–470, 2013.

44. Garca, S., J. Luengo and F. Herrera, *Data Preprocessing in Data Mining*, Springer Publishing Company Inc., 2014.

45. Pyle, D., *Data Preparation for Data Mining*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edn., 1999.

46. Jia, H. and Y. Cheung, "Subspace Clustering of Categorical and Numerical Data with an Unknown Number of Clusters", *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 29, No. 8, pp. 3308–3325, 2018.

47. Zhang, Y. and D.-Y. Yeung, "Semi-supervised Multi-task Regression", W. Buntine, M. Grobelnik, D. Mladenić and J. Shawe-Taylor (Editors), *Machine Learning and Knowledge Discovery in Databases*, pp. 617–631, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

48. Zhou, Z. H., "When Semi-supervised Learning Meets Ensemble Learning", *Frontiers in Electrical and Electronic Engineering in China*, Vol. 5, No. 3, pp. 6–16, 2011.

49. Yu, Z., Y. Zhang, C. L. P. Chen, J. You, H.-S. Wong, D. Dai, S. Wu and J. Zhang, "Multiobjective Semisupervised Classifier Ensemble", *IEEE Transactions on Cybernetics*, Vol. 49, pp. 2280–2293, 2019.

50. Blockeel, H., L. D. Raedt and J. Ramon, "Top-down Induction of Clustering Trees", *Proceedings of the 15th International Conference on Machine Learning*, p. 55–63, 1998.

51. Shi, T. and S. Horvath, "Unsupervised Learning with Random Forest Predictors", *Journal of Computational and Graphical Statistics*, Vol. 15, No. 1, pp. 118–138, 2006.

52. Basak, J. and R. Krishnapuram, "Interpretable Hierarchical Clustering by Constructing an Unsupervised Decision Tree", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, No. 1, pp. 121–132, 2006.

53. Ženko, B., S. Džeroski and J. Struyf, "Learning Predictive Clustering Rules",

F. Bonchi and J.-F. Boulicaut (Editors), *Knowledge Discovery in Inductive Databases*, pp. 234–250, Springer Berlin Heidelberg, 2006.

54. Li, M. and Z.-H. Zhou, "Improve Computer-aided Diagnosis with Machine Learning Techniques Using Undiagnosed Samples", *Proceedings of Neural Information Processing Systems*, Vol. 37, No. 6, pp. 1088–1098, 2007.

55. Leistner, C., A. Saffari, J. Santner and H. Bischof, "Semi-supervised Random Forests", *IEEE 12th International Conference on Computer Vision*, pp. 364–374, 2009.

56. Liu, X., M. Song, D. Tao, Z. Liu, L. Zhang, C. Chen and J. Bu, "Random Forest Construction with Robust Semisupervised Node Splitting", *IEEE Transactions on Image Processing*, Vol. 24, No. 1, pp. 471–482, 2015.

57. Tanha, J., M. V. Someren and H. Afsarmanesh, "Semi-supervised Self-training for Decision Tree Classifiers", *International Journal of Machine Learning and Cybernetics*, Vol. 8, pp. 355–370, 2017.

58. Kobayashi, A., T. Oku, T. Imai and S. Nakagawa, "Multi-objective Optimization for Semi-supervised Discriminative Language Modeling", *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4997–5000, 2012.

59. Levatić, J., M. Ceci, D. Kocev and S. Džeroski, "Self-training for Multi-target Regression with Tree Ensembles", *Knowledge-Based Systems*, Vol. 123, pp. 41 – 60, 2017.

60. Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, Vol. 12, pp. 2825–2830, 2011.

61. Sheikhpour, R., M. A. Sarram, S. Gharaghani and M. A. Z. Chahooki, "A Survey on Semi-supervised feature Selection Methods", *Pattern Recognition*, Vol. 64, pp. 141–158, 2017.

62. Kashef, S., H. Nezamabadi-pour and B. Nikpour, "Multilabel Feature Selection: A Comprehensive Review and Guiding Experiments", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Vol. 8, No. 2, p. e1240, 2018.

63. Petković, M., D. Kocev and S. Džeroski, "Feature Ranking for Multi-Target Regression", *Machine Learning*, Vol. 109, No. 6, pp. 1179–1204, 2020.

64. Cai, J., J. Luo, S. Wang and S. Yang, "Feature Selection in Machine Learning: A New Perspective", *Neurocomputing*, Vol. 300, pp. 70 –79, 2018.

65. Xu, Z., I. King, M. R. Lyu and R. Jin, "Discriminative Semi-supervised Feature Selection via Manifold Regularization", *IEEE Transactions on Neural Networks*, Vol. 21, No. 7, pp. 1033–1047, 2010.

66. Bellal, F., H. Elghazel and A. Aussem, "A Semi-supervised Feature Ranking Method with Ensemble Learning", *Pattern Recognition Letters*, Vol. 33, pp. 1426–1433, 2012.

67. Shi, C., Q. Ruan and G. An, "Sparse Feature Selection Based on Graph Laplacian for Web Image Annotation", *Image and Vision Computing*, Vol. 32, No. 3, pp. 189 – 201, 2014.

68. Bolón-Canedo, V. and A. Alonso-Betanzos, "Ensembles for Feature Selection: A Review and Future Trends", *Information Fusion*, Vol. 52, pp. 1–12, 2019.

69. Han, Y., K. Park and Y.-K. Lee, "Confident Wrapper-type Semi-supervised Feature Selection Using an Ensemble Classifier", *2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC)*, pp. 4581–4586, IEEE, 2011.

70. Grabner, H., C. Leistner and H. Bischof, "Semi-supervised On-line Boosting for Robust Tracking", *European Conference on Computer Vision*, pp. 234–247, Springer, 2008.

71. Settouti, N., M. A. Chikh and V. Barra, "A New Feature Selection Approach Based on Ensemble Methods in Semi-supervised Classification", *Pattern Analysis and Applications*, Vol. 20, No. 3, pp. 673–686, 2017.

72. Chang, X., F. Nie, Y. Yang and H. Huang, "A Convex Formulation for Semi-supervised Multi-label Feature Selection", *28^{th} AAAI Conference on Artificial Intelligence*, p. 1171–1177, 2014.

73. Qian, B. and I. Davidson, "Semi-supervised Dimension Reduction for Multi-label Classification", *AAAI*, Vol. 10, pp. 569–574, 2010.

74. Audet, C., J. Bigeon, D. Cartier, S. L. Digabel and S. L, *Performance Indicators in Multiobjective Optimization*, 2018, `http://www.optimization-online.org/DB_HTML/2018/10/6887.html`, Accessed on: May 2020.

75. Gong, D., Y. Liu and G. G. Yen, "A Meta-Objective Approach for Many-Objective Evolutionary Optimization", *Evolutionary Computation*, Vol. 28, No. 1, pp. 1–25, 2020.

APPENDIX A: EXPERIMENTAL RESULTS

Experimental results are given in Table A.1, Table A.2, Table A.3 and Table A.4.

Table A.1. Relative root mean squared errors of the methods.

	PMORF	**CDMORF**	**RMORF**	**SC**	**ERC**	**ST**	**MTSC**	**MORF**
Andro	0.6235	0.6228	0.5573	0.5591	0.6624	0.6944	0.6837	0.5963
y_1	0.5621	0.5370	0.4704	0.5317	0.6697	0.6800	0.6890	0.5521
y_2	0.5266	0.4874	0.4781	0.3871	0.3440	0.3523	0.3561	0.4086
y_3	0.5449	0.5897	0.5014	0.4505	0.5519	0.6021	0.5458	0.5141
y_4	0.5428	0.5845	0.5033	0.5058	0.5542	0.6078	0.5552	0.5191
y_5	0.7887	0.7763	0.7086	0.7341	0.9618	0.9861	1.0175	0.8039
y_6	0.7756	0.7622	0.6820	0.7455	0.8926	0.9382	0.9386	0.7801
ATP1D	0.4352	0.4294	0.4288	0.3803	0.3858	0.3782	0.3786	0.4183
y_1	0.4795	0.4697	0.4758	0.4718	0.4685	0.4692	0.4678	0.4768
y_2	0.4675	0.4519	0.4680	0.4358	0.4437	0.4377	0.4438	0.4641
y_3	0.4394	0.4376	0.4410	0.4304	0.4353	0.4351	0.4303	0.4330
y_4	0.3817	0.3799	0.3621	0.2608	0.2706	0.2555	0.2544	0.3391
y_5	0.4677	0.4568	0.4712	0.4519	0.4558	0.4555	0.4558	0.4706
y_6	0.3753	0.3807	0.3548	0.2313	0.2410	0.2162	0.2197	0.3261
ATP7D	0.6046	0.5981	0.5755	0.5366	0.5693	0.5629	0.5570	0.5878
y_1	0.7680	0.7540	0.7197	0.7606	0.7582	0.7710	0.7605	0.7620
y_2	0.6153	0.6216	0.6106	0.6397	0.6667	0.6601	0.6646	0.6347
y_3	0.5636	0.5592	0.5522	0.5632	0.5777	0.5820	0.5797	0.5719
y_4	0.4885	0.4792	0.4505	0.2891	0.3559	0.3420	0.3199	0.4306
y_5	0.7014	0.6933	0.6729	0.6940	0.7264	0.7193	0.7292	0.7050
y_6	0.4909	0.4814	0.4470	0.2729	0.3307	0.3033	0.2880	0.4228
CalHouse	0.6610	0.6628	0.6676	0.6196	0.6154	0.6296	0.6438	0.6492
y_1	0.6719	0.6715	0.6802	0.6452	0.6411	0.6569	0.6869	0.6717
y_2	0.6501	0.6542	0.6550	0.5941	0.5896	0.6024	0.6008	0.6267
EDM	0.7166	0.7210	0.7232	0.7435	0.7319	0.7612	0.7620	0.7202
y_1	0.7100	0.7063	0.7085	0.7663	0.7656	0.8143	0.8145	0.7322
y_2	0.7231	0.7356	0.7380	0.7207	0.6981	0.7080	0.7095	0.7082
ENB	0.1504	0.1515	0.1518	0.1285	0.1168	0.1172	0.1211	0.1218
y_1	0.1091	0.1105	0.1111	0.0725	0.0561	0.0536	0.0643	0.0597
y_2	0.1916	0.1926	0.1925	0.1845	0.1776	0.1807	0.1778	0.1839

Table A.1. Relative root mean squared errors of the methods (continued).

	PMORF	**CDMORF**	**RMORF**	**SC**	**ERC**	**ST**	**MTSC**	**MORF**
Jura	0.6049	0.6146	0.6099	0.5731	0.5903	0.5860	0.5880	0.5930
y_1	0.6747	0.6788	0.6709	0.6765	0.7032	0.7013	0.7032	0.6847
y_2	0.5416	0.5401	0.5445	0.5086	0.5318	0.5262	0.5274	0.5465
y_3	0.5984	0.6250	0.6144	0.5342	0.5361	0.5305	0.5333	0.5477
M5Spec	0.7078	0.7044	0.7060	0.6645	0.6642	0.7025	0.7005	0.7172
y_1	0.7095	0.7060	0.7077	0.6554	0.6770	0.7178	0.7255	0.7247
y_2	0.7086	0.7051	0.7067	0.6671	0.6608	0.6994	0.6938	0.7174
y_3	0.7053	0.7019	0.7035	0.6709	0.6548	0.6904	0.6822	0.7094
MP5Spec	0.6721	0.5909	0.6733	0.6433	0.6036	0.6109	0.6174	0.6380
y_1	0.6724	0.5925	0.6736	0.6422	0.6015	0.6103	0.6185	0.6382
y_2	0.6691	0.5856	0.6703	0.6428	0.6029	0.6087	0.6142	0.6356
y_3	0.6746	0.5950	0.6759	0.6432	0.6066	0.6133	0.6190	0.6399
y_4	0.6721	0.5903	0.6734	0.6449	0.6033	0.6114	0.6179	0.6384
MP6Spec	0.6680	0.6697	0.6674	0.6271	0.6257	0.6113	0.6402	0.6384
y_1	0.6662	0.6679	0.6656	0.6262	0.6203	0.6100	0.6384	0.6369
y_2	0.6670	0.6687	0.6664	0.6358	0.6308	0.6100	0.6390	0.6373
y_3	0.6703	0.6721	0.6697	0.6229	0.6298	0.6133	0.6425	0.6403
y_4	0.6685	0.6703	0.6679	0.6233	0.6219	0.6120	0.6410	0.6389
Polymer	0.6750	0.7126	0.6379	0.6914	0.6082	0.6335	0.6553	0.6418
y_1	0.7899	0.8444	0.7260	0.7147	0.7050	0.6372	0.7547	0.6931
y_2	0.8714	0.9066	0.7907	0.9606	0.7379	0.7591	0.7427	0.7872
y_3	0.5520	0.5854	0.5452	0.5678	0.4997	0.5818	0.5775	0.5643
y_4	0.4868	0.5138	0.4896	0.5226	0.4903	0.5558	0.5461	0.5224
Stock	0.7335	0.7359	0.7594	0.7082	0.7060	0.7075	0.7145	0.7232
y_1	0.8472	0.8503	0.8565	0.8483	0.8213	0.8379	0.8528	0.8293
y_2	0.7860	0.7880	0.7981	0.7797	0.7668	0.7674	0.7759	0.7708
y_3	0.5672	0.5695	0.6238	0.4968	0.5299	0.5172	0.5148	0.5695
Slump	0.2596	0.2641	0.2660	0.2701	0.3039	0.3036	0.3163	0.2942
y_1	0.2277	0.2263	0.2332	0.2398	0.2710	0.2722	0.2794	0.2388
y_2	0.3135	0.3210	0.3191	0.3315	0.3616	0.3653	0.3720	0.3547
y_3	0.2377	0.2450	0.2457	0.2390	0.2792	0.2733	0.2974	0.2892

Table A.1. Relative root mean squared errors of the methods (continued).

	PMORF	**CDMORF**	**RMORF**	**SC**	**ERC**	**ST**	**MTSC**	**MORF**
OES97	0.4134	0.4225	0.4156	0.3956	0.4144	0.4106	0.4110	0.4289
y_1	0.4041	0.4196	0.4077	0.4011	0.4333	0.4359	0.4363	0.4310
y_2	0.3514	0.3779	0.3476	0.3586	0.3806	0.3887	0.3869	0.3735
y_3	0.3847	0.4038	0.3771	0.3853	0.3935	0.4025	0.4023	0.3879
y_4	0.4617	0.4530	0.4656	0.4378	0.4410	0.4316	0.4330	0.4641
y_5	0.4574	0.4496	0.4542	0.3869	0.4320	0.4178	0.4188	0.4562
y_6	0.5690	0.5686	0.5773	0.5991	0.5913	0.5857	0.5864	0.6225
y_7	0.3506	0.3737	0.3498	0.3174	0.3421	0.3276	0.3276	0.3543
y_8	0.3644	0.3904	0.3625	0.3632	0.4014	0.4060	0.4060	0.3928
y_9	0.6063	0.6093	0.5964	0.6004	0.6070	0.6148	0.6150	0.6285
y_{10}	0.3547	0.3771	0.3503	0.3124	0.3304	0.3235	0.3247	0.3554
y_{11}	0.4471	0.4286	0.4643	0.3394	0.3965	0.3678	0.3685	0.4477
y_{12}	0.2305	0.2660	0.2274	0.2332	0.2647	0.2714	0.2721	0.2508
y_{13}	0.5571	0.5257	0.5838	0.5016	0.5049	0.4904	0.4905	0.5654
y_{14}	0.2252	0.2581	0.2241	0.2246	0.2642	0.2661	0.2671	0.2566
y_{15}	0.4432	0.4567	0.4508	0.4502	0.4553	0.4540	0.4536	0.4555
y_{16}	0.4063	0.4020	0.4113	0.4184	0.3926	0.3863	0.3869	0.4202
OES10	0.5278	0.5188	0.5335	0.5026	0.5248	0.5248	0.5253	0.5581
y_1	0.2747	0.2780	0.2798	0.2791	0.3726	0.3739	0.3721	0.3192
y_2	0.4741	0.4340	0.4765	0.3722	0.3851	0.3841	0.3881	0.4979
y_3	0.6804	0.6854	0.6892	0.6842	0.6791	0.6797	0.6794	0.6972
y_4	0.3879	0.4015	0.3965	0.3550	0.3778	0.3772	0.3785	0.4079
y_5	0.7023	0.6687	0.7084	0.6377	0.6746	0.6718	0.6775	0.7246
y_6	0.6598	0.6373	0.6762	0.5833	0.6005	0.6013	0.6006	0.6897
y_7	0.6007	0.6008	0.6094	0.5845	0.5707	0.5704	0.5712	0.6300
y_8	0.2734	0.2806	0.2642	0.2761	0.3523	0.3529	0.3518	0.3064
y_9	0.2570	0.2927	0.2478	0.2409	0.3113	0.3108	0.3117	0.2581
y_{10}	0.6082	0.5727	0.6163	0.5509	0.5582	0.5579	0.5597	0.6355

Table A.1. Relative root mean squared errors of the methods (continued).

	PMORF	**CDMORF**	**RMORF**	**SC**	**ERC**	**ST**	**MTSC**	**MORF**
y_{11}	0.6056	0.5965	0.6128	0.6629	0.6443	0.6439	0.6448	0.6683
y_{12}	0.5010	0.4857	0.5166	0.5228	0.5275	0.5283	0.5274	0.5567
y_{13}	0.5469	0.5487	0.5476	0.5426	0.5538	0.5541	0.5533	0.5579
y_{14}	0.7086	0.6554	0.7369	0.6235	0.6573	0.6586	0.6561	0.7640
y_{15}	0.5797	0.5814	0.5760	0.5600	0.5617	0.5616	0.5628	0.6040
y_{16}	0.5838	0.5823	0.5818	0.5656	0.5697	0.5697	0.5704	0.6125
Puma8NH	0.7768	0.7907	0.7780	0.7978	0.8131	0.8149	0.8269	0.7962
y_1	0.9033	0.9019	0.9062	0.9109	0.9553	0.9582	0.9736	0.9344
y_2	0.8977	0.8970	0.9062	0.9520	0.9553	0.9569	0.9716	0.9351
y_3	0.5293	0.5731	0.5216	0.5306	0.5288	0.5296	0.5356	0.5189
Water Quality	0.9115	0.9126	0.9096	0.9209	0.9130	0.9154	0.9182	0.9124
y_1	0.9284	0.9324	0.9306	0.9361	0.9286	0.9302	0.9337	0.9320
y_2	0.9795	0.9773	0.9803	1.0008	0.9842	0.9876	0.9900	0.9802
y_3	0.9453	0.9563	0.9504	0.9699	0.9513	0.9549	0.9594	0.9565
y_4	0.9117	0.9070	0.9127	0.9320	0.9126	0.9189	0.9166	0.9165
y_5	0.9126	0.9138	0.9044	0.9195	0.9136	0.9127	0.9227	0.9148
y_6	0.8349	0.8385	0.8348	0.8410	0.8379	0.8412	0.8421	0.8353
y_7	0.9620	0.9586	0.9615	0.9898	0.9689	0.9716	0.9724	0.9700
y_8	0.9160	0.9192	0.9183	0.9269	0.9168	0.9183	0.9196	0.9159
y_9	0.8283	0.8267	0.8153	0.8157	0.8189	0.8196	0.8259	0.8174
y_{10}	0.9038	0.9052	0.9026	0.8952	0.8977	0.8976	0.9019	0.8998
y_{11}	0.9232	0.9203	0.9127	0.9227	0.9272	0.9277	0.9334	0.9227
y_{12}	0.9048	0.9073	0.9046	0.9228	0.9200	0.9232	0.9230	0.9119
y_{13}	0.9497	0.9551	0.9499	0.9547	0.9522	0.9547	0.9590	0.9484
y_{14}	0.8616	0.8589	0.8567	0.8651	0.8525	0.8572	0.8549	0.8528
Wisconsin Cancer	0.9503	0.9430	0.9453	0.9341	0.9141	0.9104	0.9133	0.9079
y_1	0.9403	0.9216	0.9246	0.8952	0.8747	0.8693	0.8719	0.8807
y_2	0.9603	0.9644	0.9660	0.9731	0.9534	0.9515	0.9546	0.9351

Table A.2. Relative root mean squared errors of transductive methods.

			5%					**10%**					**20%**		
	PCT	**PCTF**	**ERF**	**TRF**	**R-TRF**	**PCT**	**PCTF**	**ERF**	**TRF**	**R-TRF**	**PCT**	**PCTF**	**ERF**	**TRF**	**R-TRF**
WQ-y_1	1.0000	0.9797	0.9730	0.9843	0.9815	1.0000	0.9637	0.9788	0.9653	0.9712	1.0118	0.9510	0.9531	0.9537	0.9557
y_2	1.0000	10127	0.9949	1.0244	1.0258	1.0000	1.0109	1.0113	1.0199	1.0211	1.0171	0.9985	1.0065	1.0057	1.0059
y_3	1.0000	0.9805	1.0001	0.9860	0.9905	1.0000	0.9777	1.0040	0.9829	0.9910	1.0070	0.9708	0.9760	0.9726	0.9772
y_4	1.0000	0.9589	1.0145	0.9580	0.9581	1.0000	0.9445	0.9523	0.9360	0.9428	1.0124	0.9240	0.9167	0.9186	0.9206
y_5	1.0000	0.9779	0.9853	0.9791	0.9845	1.0000	0.9521	0.9287	0.9493	0.9609	1.0109	0.9428	0.9423	0.9425	0.9498
y_6	1.0000	0.8808	0.8804	0.8781	0.8816	1.0000	0.8746	0.9229	0.8755	0.8803	1.0130	0.8584	0.8570	0.8587	0.8597
y_7	1.0000	0.9957	1.0116	1.0010	1.0061	1.0000	0.9812	0.9604	0.9845	0.9863	1.0066	0.9739	0.9769	0.9784	0.9738
y_8	1.0000	0.9534	0.9651	0.9560	0.9577	1.0000	0.9357	0.9450	0.9387	0.9479	1.0113	0.9266	0.9298	0.9274	0.9337
y_9	1.0000	0.9383	0.9383	0.9258	0.9350	1.0000	0.9058	0.8802	0.9026	0.9075	1.0190	0.8791	0.8753	0.8789	0.8832
y_{10}	1.0000	0.9942	0.9864	0.9926	0.9976	1.0000	0.9625	1.0150	0.9593	0.9656	1.0100	0.9388	0.9407	0.9385	0.9461
y_{11}	1.0000	0.9778	0.9955	0.9700	0.9794	1.0000	0.9678	0.9612	0.9572	0.9659	1.0157	0.9497	0.9451	0.9457	0.9476
y_{12}	1.0000	0.9569	0.9446	0.9521	0.9667	1.0000	0.9547	0.9763	0.9465	0.9566	1.0046	0.9261	0.9222	0.9164	0.9255
y_{13}	1.0000	1.0063	1.0191	1.0006	1.0062	1.0000	0.9891	0.9837	0.9917	0.9926	1.0114	0.9891	0.9873	0.9899	0.9874
y_{14}	1.0000	0.8936	0.8680	0.8915	0.9022	1.0000	0.8757	0.9155	0.8751	0.8802	1.0079	0.8610	0.8635	0.8639	0.8680
MP6- y_1	0.9074	0.8461	0.7453	0.7499	0.7735	0.7640	0.7157	0.6583	0.6640	0.6923	0.6774	0.6148	0.5836	0.5962	0.6082
y_2	0.9081	0.8470	0.7470	0.7517	0.7753	0.7651	0.7171	0.6595	0.6655	0.6938	0.6786	0.6160	0.5849	0.5976	0.6096
y_3	0.9101	0.8497	0.7522	0.7572	0.7810	0.7690	0.7214	0.6643	0.6701	0.6985	0.6825	0.6201	0.5890	0.6015	0.6139
y_4	0.9090	0.8483	0.7494	0.7543	0.7780	0.7670	0.7192	0.6618	0.6677	0.6960	0.6805	0.6180	0.5868	0.5995	0.6116
MP5- y_1	0.8550	0.7826	0.7160	0.7154	0.7101	0.8061	0.7996	0.8525	0.6642	0.6659	0.6929	0.6175	0.5690	0.5763	0.5878
y_2	0.8509	0.7761	0.7092	0.7086	0.7035	0.8005	0.7937	0.8409	0.6576	0.6595	0.6870	0.6117	0.5644	0.5716	0.5828
y_3	0.8570	0.7850	0.7185	0.7179	0.7128	0.8085	0.8017	0.8562	0.6666	0.6684	0.6955	0.6199	0.5713	0.5787	0.5902
y_4	0.8543	0.7808	0.7141	0.7135	0.7086	0.8048	0.7980	0.8489	0.6625	0.6644	0.6917	0.6162	0.5683	0.5756	0.5870
OES10-y_1	0.8122	0.7366	0.6928	0.6899	0.6746	0.7149	0.6459	0.6508	0.6130	0.5934	0.6305	0.6424	0.5630	0.5711	0.5470
y_2	0.7637	0.6977	0.6398	0.6400	0.6100	0.6968	0.5811	0.5874	0.5268	0.5034	0.6800	0.5788	0.4816	0.4908	0.4605
y_3	0.7805	0.7173	0.6519	0.6552	0.6307	0.7039	0.6117	0.6072	0.5813	0.5609	0.8366	0.6093	0.4994	0.5104	0.4842
y_4	0.8395	0.7799	0.7466	0.7571	0.7326	0.7497	0.7187	0.6916	0.6860	0.6652	0.5982	0.7136	0.6065	0.6112	0.5813
y_5	0.8937	0.8088	0.7688	0.7726	0.7505	0.7805	0.7055	0.6903	0.6576	0.6348	0.9251	0.7086	0.6137	0.6186	0.5883
y_6	0.8928	0.8526	0.8324	0.8316	0.8260	0.8300	0.8115	0.6701	0.7896	0.7846	0.7986	0.8038	0.7513	0.7536	0.7316
y_7	0.8377	0.7597	0.7189	0.7204	0.6895	0.7443	0.6876	0.7472	0.6418	0.6220	0.7504	0.6805	0.5961	0.6014	0.5570
y_8	0.7886	0.7096	0.6588	0.6608	0.6369	0.6982	0.6338	0.6351	0.6013	0.5815	0.6272	0.6284	0.5518	0.5588	0.5294
y_9	0.9343	0.8319	0.8131	0.8184	0.7942	0.8332	0.7611	0.8634	0.7323	0.7125	0.6140	0.7639	0.6873	0.6923	0.6726
y_{10}	0.7665	0.6729	0.6173	0.6170	0.5960	0.6849	0.5775	0.5818	0.5216	0.5067	0.7749	0.5789	0.4659	0.4721	0.4622
y_{11}	0.7919	0.7005	0.6552	0.6603	0.6288	0.7235	0.6254	0.6553	0.5935	0.5769	0.7821	0.6256	0.5754	0.5837	0.5487
y_{12}	0.7585	0.6486	0.5861	0.5829	0.5761	0.6312	0.5564	0.5210	0.4871	0.4776	0.7077	0.5501	0.4378	0.4447	0.4200
y_{13}	0.8935	0.8063	0.7927	0.7991	0.7823	0.8193	0.7721	0.6596	0.7473	0.7429	0.7110	0.7369	0.7073	0.6923	0.7051
y_{14}	0.7761	0.7015	0.6445	0.6447	0.6226	0.6822	0.6203	0.5886	0.5679	0.5461	0.8379	0.6066	0.5343	0.5392	0.5007
y_{15}	0.8294	0.7600	0.7188	0.7216	0.7004	0.7225	0.6573	0.5857	0.6068	0.6031	0.7581	0.6493	0.5513	0.5540	0.5455
y_{16}	0.7947	0.7180	0.6705	0.6676	0.6559	0.6982	0.6405	0.5543	0.5995	0.5903	0.7517	0.6337	0.5789	0.5815	0.5664

Table A.2. Relative root mean squared errors of transductive methods (continued).

			5%					**10%**					**20%**		
	PCT	**PCTF**	**ERF**	**TRF**	**R-TRF**	**PCT**	**PCTF**	**ERF**	**TRF**	**R-TRF**	**PCT**	**PCTF**	**ERF**	**TRF**	**R-TRF**
OES97- y_1	0.8170	0.7422	0.7038	0.7227	0.6739	0.7249	0.6741	0.6850	0.6343	0.6182	0.6305	0.5618	0.5150	0.5196	0.5134
y_2	0.7995	0.7729	0.7455	0.7398	0.7108	0.7664	0.7224	0.7399	0.6977	0.6813	0.6800	0.6053	0.5789	0.5803	0.5825
y_3	0.9934	0.8829	0.8643	0.9098	0.8569	0.9082	0.8339	0.8294	0.8072	0.8019	0.8366	0.7559	0.7441	0.7536	0.7393
y_4	0.7625	0.7636	0.7243	0.7172	0.6967	0.6979	0.6248	0.6762	0.5953	0.5809	0.5982	0.5028	0.4706	0.4592	0.4845
y_5	0.9383	0.8492	0.8403	0.8143	0.8200	0.9114	0.8139	1.0264	0.8115	0.7901	0.9251	0.7638	0.7465	0.7379	0.7367
y_6	0.8909	0.8663	0.8485	0.8324	0.8289	0.8362	0.7769	0.8129	0.7630	0.7442	0.7986	0.7026	0.6864	0.6819	0.6752
y_7	0.8414	0.8370	0.8037	0.8113	0.7813	0.8073	0.7733	0.7748	0.7464	0.7333	0.7504	0.6352	0.6460	0.6532	0.6351
y_8	0.8328	0.7651	0.7250	0.7378	0.6972	0.7343	0.6876	0.6697	0.6440	0.6228	0.6272	0.5389	0.4927	0.5103	0.4935
y_9	0.8458	0.7343	0.6904	0.7488	0.6633	0.7082	0.6425	0.6360	0.5910	0.5710	0.6140	0.5081	0.4533	0.4601	0.4592
y_{10}	0.8381	0.8248	0.8007	0.7892	0.7755	0.8116	0.7719	0.8082	0.7512	0.7337	0.7749	0.7003	0.6818	0.6898	0.6803
y_{11}	0.8763	0.8371	0.8057	0.8008	0.7867	0.8685	0.7976	0.8891	0.7729	0.7614	0.7821	0.7104	0.6903	0.7190	0.6828
y_{12}	0.8262	0.8059	0.7752	0.7561	0.7535	0.7871	0.7358	0.8140	0.7215	0.7110	0.7077	0.6427	0.6077	0.6121	0.6072
y_{13}	0.8365	0.7842	0.7546	0.7242	0.7328	0.7723	0.6792	0.6618	0.6515	0.6308	0.7110	0.6057	0.5756	0.5751	0.5761
y_{14}	0.9311	0.8620	0.8505	0.8475	0.8276	0.8811	0.8625	0.9558	0.8467	0.8375	0.8379	0.7937	0.7667	0.7698	0.7672
y_{15}	0.9328	0.8157	0.7898	0.8475	0.7692	0.8412	0.7399	0.8015	0.7091	0.6841	0.7581	0.6520	0.6401	0.6534	0.6391
y_{16}	0.9454	0.8300	0.7960	0.8577	0.7735	0.8403	0.7761	0.7575	0.7498	0.7325	0.7517	0.6867	0.6675	0.6786	0.6648
ENB - y_1	0.3140	0.2711	0.2926	0.2958	0.2958	0.2322	0.2123	0.2576	0.2247	0.2247	0.1422	0.1549	0.2018	0.2012	0.2012
y_2	0.3344	0.3030	0.3245	0.3252	0.3252	0.2947	0.2535	0.3143	0.2648	0.2648	0.2400	0.2127	0.2486	0.2475	0.2475
CALHOUSE - y_1	10211	0.9657	0.9539	0.9465	0.9465	0.9957	0.8926	0.8872	0.8927	0.8927	10157	0.8188	0.8252	0.8299	0.8299
y_2	10360	0.9380	0.9332	0.9333	0.9333	10208	0.8651	0.8817	0.8772	0.8772	0.9955	0.7907	0.8161	0.8206	0.8206

Table A.3. Relative root mean squared errors of inductive methods.

			5%					10%					20%		
	PCT	**PCTF**	**ERF**	**TRF**	**RTRF**	**PCT**	**PCTF**	**ERF**	**TRF**	**RTRF**	**PCT**	**PCTF**	**ERF**	**TRF**	**R-TRF**
WQ - y_1	1.0246	1.0016	0.9877	0.9842	0.9832	1.0118	0.9848	0.9788	0.9708	0.9760	1.0258	0.9650	0.9859	0.9646	0.9574
y_2	1.0233	1.0393	1.0079	1.0216	1.0247	1.0174	1.0261	1.0113	1.0028	1.0151	1.0308	1.0230	1.0188	1.0021	1.0101
y_3	1.0165	1.0016	0.9956	0.9933	1.0057	1.0132	0.9958	1.0040	0.9911	0.9955	1.0144	0.9770	0.9701	0.9737	0.9895
y_4	1.0107	0.9878	1.0197	0.9732	0.9673	1.0066	0.9499	0.9523	0.9354	0.9412	1.0477	0.9397	0.8936	0.9295	0.9171
y_5	1.0097	1.0001	0.9853	0.9859	1.0035	1.0050	0.9538	0.9287	0.9444	0.9529	1.0163	0.9481	0.9054	0.9364	0.9557
y_6	1.0119	0.9040	0.9567	0.8854	0.8928	1.0101	0.8792	0.9229	0.8722	0.8765	1.0297	0.8619	0.9388	0.8536	0.8570
y_7	1.0058	0.9942	1.0028	0.9905	0.9969	1.0099	0.9841	0.9604	0.9775	0.9785	1.0131	0.9871	0.9258	0.9728	0.9797
y_8	1.0158	0.9652	0.9074	0.9517	0.9596	1.0196	0.9522	0.9450	0.9261	0.9334	1.0198	0.9451	0.9151	0.9306	0.9250
y_9	1.0181	0.9480	0.8658	0.9251	0.9259	1.0136	0.9151	0.8802	0.8935	0.9010	1.0267	0.8855	0.8454	0.8763	0.8765
y_{10}	1.0182	1.0189	1.0131	0.9980	1.0022	1.0110	0.9786	1.0150	0.9685	0.9716	1.0096	0.9546	0.9454	0.9437	0.9494
y_{11}	1.0060	0.9964	0.9596	0.9944	0.9963	1.0040	0.9763	0.9612	0.9644	0.9747	1.0182	0.9439	0.9175	0.9424	0.9465
y_{12}	1.0093	0.9767	0.9773	0.9626	0.9760	1.0108	0.9727	0.9763	0.9494	0.9591	1.0218	0.9465	0.9637	0.9337	0.9429
y_{13}	1.0153	1.0280	1.0233	1.0043	1.0062	1.0078	0.9863	0.9837	0.9846	0.9868	1.0225	0.9859	0.9860	0.9804	0.9829
y_{14}	1.0224	0.9226	0.9528	0.9047	0.9150	1.0091	0.8952	0.9155	0.8767	0.8939	1.0318	0.8654	0.9243	0.8645	0.8717
MP6 - y_1	0.8322	0.8526	0.6589	0.6837	0.6875	0.7462	0.7238	0.6583	0.6130	0.6329	0.6891	0.6421	0.6980	0.5505	0.5739
y_2	0.8338	0.8536	0.6606	0.6851	0.6889	0.7474	0.7248	0.6595	0.6140	0.6339	0.6904	0.6432	0.6995	0.5518	0.5753
y_3	0.8386	0.8569	0.6655	0.6892	0.6928	0.7515	0.7279	0.6643	0.6177	0.6373	0.6951	0.6468	0.7059	0.5558	0.5798
y_4	0.8361	0.8552	0.6629	0.6871	0.6908	0.7494	0.7263	0.6618	0.6158	0.6356	0.6925	0.6449	0.7025	0.5537	0.5774
MP5 - y_1	0.9244	0.8679	0.8727	0.7587	0.7748	1.0153	0.9323	0.8525	0.7253	0.7376	0.7222	0.6547	0.8195	0.6236	0.6393
y_2	0.9230	0.8659	0.8669	0.7531	0.7691	1.0091	0.9264	0.8409	0.7190	0.7314	0.7176	0.6495	0.8095	0.6188	0.6336
y_3	0.9263	0.8702	0.8762	0.7614	0.7777	1.0178	0.9351	0.8562	0.7280	0.7404	0.7248	0.6570	0.8229	0.6259	0.6419
y_4	0.9251	0.8687	0.8724	0.7577	0.7739	1.0137	0.9311	0.8489	0.7240	0.7364	0.7218	0.6537	0.8162	0.6228	0.6382
OES10 - y_1	0.7995	0.7036	0.6729	0.6177	0.6090	0.6827	0.6010	0.6508	0.5583	0.5416	0.7006	0.5391	0.6929	0.5147	0.5122
y_2	0.7529	0.6688	0.6555	0.5816	0.5647	0.6468	0.5317	0.5874	0.4792	0.4674	0.6690	0.4758	0.5804	0.4568	0.4363
y_3	0.7715	0.6928	0.6696	0.6066	0.6049	0.6677	0.5818	0.6072	0.5192	0.5193	0.6635	0.5161	0.6086	0.4767	0.4731
y_4	0.7583	0.7158	0.7321	0.6832	0.6574	0.7155	0.6299	0.6916	0.6007	0.5697	0.7470	0.5448	0.6859	0.5377	0.5116
y_5	0.8802	0.7849	0.8046	0.7345	0.7133	0.7659	0.6515	0.6903	0.6032	0.5777	0.8059	0.5839	0.7178	0.5708	0.5394
y_6	0.8204	0.7406	0.7286	0.6886	0.7028	0.7367	0.6607	0.6701	0.6183	0.6240	0.7702	0.6046	0.7256	0.5954	0.5757
y_7	0.8292	0.6908	0.7612	0.6394	0.6037	0.7343	0.6117	0.7472	0.5866	0.5480	0.8150	0.5654	0.7990	0.5463	0.5086
y_8	0.8006	0.6865	0.6496	0.5760	0.5585	0.6650	0.6011	0.6351	0.5256	0.5067	0.6952	0.5258	0.6309	0.4783	0.4800
y_9	0.9575	0.8583	0.8808	0.8168	0.7946	0.7906	0.7505	0.8634	0.7306	0.6862	0.8349	0.6669	0.8267	0.6416	0.6289
y_{10}	0.7708	0.6814	0.6568	0.5849	0.5888	0.6530	0.5629	0.5818	0.5091	0.4987	0.6516	0.5030	0.5680	0.4710	0.4682
y_{11}	0.7574	0.6798	0.6321	0.5886	0.5791	0.7179	0.6172	0.6553	0.5684	0.5502	0.7335	0.5476	0.6453	0.5359	0.5133
y_{12}	0.7040	0.6084	0.5915	0.4974	0.5139	0.5456	0.4723	0.5210	0.3899	0.3962	0.5709	0.3831	0.5483	0.3531	0.3516
y_{13}	0.8941	0.7392	0.7288	0.7028	0.6904	0.7679	0.6571	0.6596	0.6222	0.6142	0.8371	0.6701	0.7261	0.6565	0.6234
y_{14}	0.7546	0.6217	0.6355	0.5248	0.5099	0.5739	0.4877	0.5886	0.4327	0.4103	0.5963	0.4027	0.5790	0.3717	0.3617
y_{15}	0.7903	0.7122	0.7194	0.6321	0.6390	0.6818	0.5831	0.5857	0.5134	0.5228	0.6905	0.5259	0.6103	0.4716	0.4738
y_{16}	0.6895	0.6108	0.5403	0.4982	0.4963	0.5652	0.5057	0.5543	0.4643	0.4502	0.6463	0.4466	0.5693	0.4444	0.4450

Table A.3. Relative root mean squared errors of inductive methods (continued).

			5%					10%					20%		
	PCT	**PCTF**	**ERF**	**TRF**	**RTRF**	**PCT**	**PCTF**	**ERF**	**TRF**	**RTRF**	**PCT**	**PCTF**	**ERF**	**TRF**	**R-TRF**
OES97 - y_1	0.7764	0.6461	0.7296	0.5629	0.5331	0.9782	0.5755	0.6850	0.4688	0.4463	0.6005	0.4255	0.5815	0.3732	0.3526
y_2	0.7432	0.7179	0.7339	0.6599	0.6522	0.9817	0.6897	0.7399	0.6292	0.6291	0.6913	0.5854	0.7163	0.5527	0.5385
y_3	1.0038	0.8780	0.8714	0.8263	0.8371	1.0456	0.8370	0.8294	0.7843	0.7871	0.9057	0.7582	0.8263	0.7153	0.7062
y_4	0.8131	0.7443	0.7304	0.6633	0.6528	0.9174	0.6875	0.6762	0.5915	0.5871	0.6443	0.5404	0.6559	0.4824	0.4948
y_5	1.1425	0.8855	0.9578	0.8247	0.8059	1.4654	0.9667	10264	0.8758	0.8422	1.1243	0.8648	0.9400	0.7861	0.7601
y_6	0.9906	0.9050	0.8926	0.8488	0.8513	1.0028	0.8257	0.8129	0.7780	0.7725	0.9065	0.7707	0.7858	0.7457	0.7203
y_7	0.8951	0.8097	0.8042	0.7387	0.7263	1.0418	0.7606	0.7748	0.6909	0.6896	0.7649	0.6570	0.7780	0.5968	0.5998
y_8	0.8264	0.6706	0.7410	0.5898	0.5760	0.9532	0.5808	0.6697	0.4900	0.4799	0.6153	0.4217	0.6341	0.3835	0.3657
y_9	0.8509	0.7057	0.7077	0.6075	0.5901	0.8025	0.6118	0.6360	0.5263	0.5036	0.6101	0.4380	0.5488	0.3649	0.3707
y_{10}	0.8311	0.7929	0.8234	0.7414	0.7352	1.1016	0.7371	0.8082	0.6969	0.6923	0.8073	0.6856	0.8071	0.6630	0.6223
y_{11}	0.9740	0.8063	0.8415	0.7406	0.7213	1.3308	0.7602	0.8891	0.7563	0.7604	1.0705	0.7873	0.9286	0.7528	0.7196
y_{12}	0.9022	0.7367	0.7845	0.6831	0.6828	1.0994	0.7597	0.8140	0.6736	0.6519	0.8562	0.6848	0.7786	0.6089	0.5916
y_{13}	0.8476	0.7797	0.7574	0.7239	0.7036	0.9541	0.6775	0.6618	0.6125	0.6201	0.7259	0.6112	0.6716	0.5669	0.5627
y_{14}	0.9611	0.8615	0.8803	0.8345	0.8241	11277	0.9051	0.9558	0.8524	0.8513	0.8884	0.7754	0.8235	0.7577	0.7298
y_{15}	10206	0.8865	0.8963	0.7854	0.7606	10990	0.7683	0.8015	0.6895	0.6735	0.7783	0.7137	0.8031	0.6228	0.6227
y_{16}	0.9782	0.8521	0.8253	0.7632	0.7371	0.9681	0.7592	0.7575	0.6815	0.6684	0.8126	0.6651	0.7337	0.6055	0.6001
ENB - y_1	0.4313	0.2632	0.3241	0.2833	0.2833	0.3490	0.2063	0.2576	0.2264	0.2264	0.2759	0.1544	0.2018	0.1955	0.1955
y_2	0.4541	0.3016	0.3704	0.3234	0.3234	0.3991	0.2513	0.3143	0.2672	0.2672	0.3518	0.2124	0.2711	0.2399	0.2399
CALHOUSE - y_1	10403	0.9782	0.9287	0.9604	0.9604	1.0093	0.8958	0.8872	0.8900	0.8900	0.9980	0.8031	0.9673	0.8034	0.8034
y_2	1.0341	0.9679	0.9619	0.9455	0.9455	1.0437	0.8810	0.8817	0.8786	0.8786	0.9835	0.7859	0.8483	0.8064	0.8064

Table A.4. Relative root mean squared errors of for feature ranking strategies.

			%5					%10					%20		
	SSS	**No FR**	**Symbolic**	**Genie3**	**RFI**	**SSS**	**No FR**	**Symbolic**	**Genie3**	**RFI**	**SSS**	**No FR**	**Symbolic**	**Genie3**	**RFI**
OES10	0.7408	0.7439	0.7389	0.7429	0.7390	0.6593	0.6629	0.6575	0.6604	0.6594	0.5785	0.6569	0.5669	0.5676	0.5684
y_1	0.7333	0.7366	0.7319	0.7301	0.7321	0.6411	0.6459	0.6412	0.6416	0.6423	0.5551	0.6424	0.5761	0.5716	0.5741
y_2	0.6934	0.6977	0.6938	0.6967	0.6924	0.5780	0.5811	0.5771	0.5827	0.5814	0.4899	0.5788	0.5056	0.5057	0.5059
y_3	0.7163	0.7173	0.7142	0.7215	0.7143	0.6072	0.6117	0.6089	0.6105	0.6098	0.5004	0.6093	0.5131	0.5208	0.5208
y_4	0.7814	0.7799	0.7762	0.7749	0.7762	0.7163	0.7187	0.7119	0.7162	0.7127	0.6105	0.7136	0.5777	0.5851	0.5816
y_5	0.8145	0.8088	0.8049	0.8059	0.8057	0.7047	0.7055	0.6970	0.7055	0.6963	0.6315	0.7086	0.6012	0.6048	0.6071
y_6	0.8546	0.8526	0.8502	0.8505	0.8516	0.8112	0.8115	0.8059	0.8074	0.8105	0.7422	0.8038	0.6734	0.6754	0.6776
y_7	0.7537	0.7597	0.7542	0.7590	0.7546	0.6793	0.6876	0.6822	0.6849	0.6823	0.5904	0.6805	0.5720	0.5791	0.5749
y_8	0.7033	0.7096	0.7010	0.7149	0.7033	0.6256	0.6338	0.6277	0.6309	0.6292	0.5442	0.6284	0.5480	0.5490	0.5505
y_9	0.8353	0.8319	0.8270	0.8505	0.8301	0.7551	0.7611	0.7530	0.7574	0.7552	0.6853	0.7639	0.6901	0.6907	0.6895
y_{10}	0.6666	0.6729	0.6676	0.6788	0.6669	0.5730	0.5775	0.5703	0.5755	0.5738	0.4916	0.5789	0.4878	0.4916	0.4906
y_{11}	0.6867	0.7005	0.6897	0.6867	0.6892	0.6287	0.6254	0.6261	0.6235	0.6260	0.5625	0.6256	0.5751	0.5721	0.5786
y_{12}	0.6396	0.6486	0.6412	0.6494	0.6389	0.5507	0.5564	0.5502	0.5531	0.5520	0.4445	0.5501	0.4123	0.4143	0.4133
y_{13}	0.8078	0.8063	0.8055	0.8035	0.8048	0.7757	0.7721	0.7676	0.7675	0.7706	0.7360	0.7369	0.6915	0.6725	0.6800
y_{14}	0.6935	0.7015	0.6932	0.6977	0.6929	0.6107	0.6203	0.6140	0.6169	0.6145	0.5222	0.6066	0.4872	0.4896	0.4906
y_{15}	0.7572	0.7600	0.7563	0.7562	0.7568	0.6546	0.6573	0.6522	0.6553	0.6552	0.5667	0.6493	0.5745	0.5783	0.5765
y_{16}	0.7158	0.7180	0.7155	0.7101	0.7150	0.6367	0.6405	0.6348	0.6381	0.6388	0.5828	0.6337	0.5842	0.5801	0.5824
OES97	0.8036	0.8108	0.8056	0.8084	0.8065	0.7128	0.7445	0.7407	0.7430	0.7427	0.6421	0.6479	0.6402	0.6458	0.6406
y_1	0.7279	0.7422	0.7334	0.7360	0.7345	0.6373	0.6741	0.6681	0.6704	0.6683	0.5497	0.5618	0.5485	0.5555	0.5488
y_2	0.7566	0.7729	0.7672	0.7843	0.7683	0.6740	0.7224	0.7192	0.7222	0.7207	0.5826	0.6053	0.5940	0.5986	0.5924
y_3	0.8819	0.8829	0.8766	0.8743	0.8806	0.8048	0.8339	0.8330	0.8329	0.8283	0.7566	0.7559	0.7504	0.7595	0.7530
y_4	0.7641	0.7636	0.7565	0.7623	0.7538	0.5770	0.6248	0.6187	0.6213	0.6292	0.4971	0.5028	0.4975	0.5089	0.5116
y_5	0.8349	0.8492	0.8478	0.8492	0.8485	0.7964	0.8139	0.8017	0.8073	0.8024	0.7665	0.7638	0.7598	0.7651	0.7582
y_6	0.8587	0.8663	0.8642	0.8641	0.8636	0.7482	0.7769	0.7794	0.7784	0.7733	0.7053	0.7026	0.7011	0.7000	0.6919
y_7	0.8369	0.8370	0.8342	0.8342	0.8330	0.7250	0.7733	0.7694	0.7723	0.7668	0.6357	0.6352	0.6350	0.6356	0.6319
y_8	0.7536	0.7651	0.7562	0.7616	0.7583	0.6436	0.6876	0.6825	0.6841	0.6859	0.5251	0.5389	0.5266	0.5359	0.5289
y_9	0.7333	0.7343	0.7261	0.7234	0.7283	0.5936	0.6425	0.6338	0.6352	0.6452	0.5003	0.5081	0.4927	0.5006	0.5009
y_{10}	0.8113	0.8248	0.8204	0.8257	0.8200	0.7590	0.7719	0.7676	0.7713	0.7650	0.6844	0.7003	0.6872	0.6984	0.6945
y_{11}	0.8276	0.8371	0.8305	0.8331	0.8326	0.7832	0.7976	0.7993	0.8011	0.7994	0.7054	0.7104	0.7062	0.7068	0.6963
y_{12}	0.7965	0.8059	0.8016	0.8030	0.8012	0.7159	0.7358	0.7353	0.7382	0.7350	0.6374	0.6427	0.6374	0.6448	0.6320
y_{13}	0.7763	0.7842	0.7779	0.7828	0.7776	0.6585	0.6792	0.6785	0.6843	0.6919	0.5968	0.6057	0.6004	0.6029	0.5972
y_{14}	0.8471	0.8620	0.8585	0.8624	0.8609	0.8343	0.8625	0.8528	0.8567	0.8544	0.7812	0.7937	0.7815	0.7880	0.7811
y_{15}	0.8241	0.8157	0.8143	0.8122	0.8159	0.7085	0.7399	0.7402	0.7390	0.7413	0.6650	0.6520	0.6491	0.6531	0.6521
y_{16}	0.8271	0.8300	0.8242	0.8257	0.8276	0.7455	0.7761	0.7712	0.7731	0.7763	0.6841	0.6867	0.6752	0.6796	0.6792

Table A.4. Relative root mean squared errors of for feature ranking strategies (continued).

			%5					%10					%20		
	SSS	**No FR**	**Symbolic**	**Genie3**	**RFI**	**SSS**	**No FR**	**Symbolic**	**Genie3**	**RFI**	**SSS**	**No FR**	**Symbolic**	**Genie3**	**RFI**
ATP1D	0.6623	0.6574	0.6497	0.6489	0.6495	0.5781	0.5765	0.5811	0.5774	0.5791	0.6561	0.6670	0.6638	0.6536	0.6635
y_1	0.7238	0.7195	0.7174	0.7100	0.7110	0.6549	0.6398	0.6526	0.6577	0.6428	0.7355	0.7573	0.7508	0.7349	0.7490
y_2	0.7264	0.7163	0.7128	0.6986	0.7033	0.6201	0.6113	0.6133	0.6140	0.6093	0.7046	0.7101	0.7139	0.7037	0.7094
y_3	0.6364	0.6331	0.6228	0.6201	0.6193	0.5599	0.5554	0.5600	0.5466	0.5606	0.6171	0.6250	0.6243	0.6146	0.6261
y_4	0.5759	0.5675	0.5560	0.5653	0.5659	0.4852	0.4922	0.4958	0.4849	0.4961	0.5458	0.5597	0.5470	0.5424	0.5509
y_5	0.7318	0.7350	0.7258	0.7236	0.7227	0.6546	0.6554	0.6575	0.6633	0.6568	0.7816	0.7862	0.7947	0.7774	0.7901
y_6	0.5792	0.5731	0.5635	0.5760	0.5745	0.4935	0.5046	0.5071	0.4978	0.5091	0.5523	0.5637	0.5520	0.5487	0.5555
ATP7D	0.8053	0.8047	0.8007	0.7974	0.8055	0.7144	0.7279	0.7701	0.7151	0.7270	0.6561	0.6670	0.6638	0.6536	0.6635
y_1	0.8801	0.8760	0.8741	0.8855	0.8755	0.8025	0.8215	0.8540	0.8207	0.8221	0.7355	0.7573	0.7508	0.7349	0.7490
y_2	0.8812	0.8730	0.8743	0.8704	0.8773	0.7562	0.7535	0.8379	0.7519	0.7644	0.7046	0.7101	0.7139	0.7037	0.7094
y_3	0.7569	0.7558	0.7480	0.7371	0.7553	0.6819	0.7035	0.7270	0.6725	0.6805	0.6171	0.6250	0.6243	0.6146	0.6261
y_4	0.7098	0.7148	0.7094	0.6983	0.7204	0.6003	0.6155	0.6600	0.5924	0.6145	0.5458	0.5597	0.5470	0.5424	0.5509
y_5	0.8892	0.8881	0.8830	0.8876	0.8803	0.8381	0.8541	0.8919	0.8551	0.8597	0.7816	0.7862	0.7947	0.7774	0.7901
y_6	0.7149	0.7205	0.7153	0.7054	0.7244	0.6077	0.6195	0.6496	0.5980	0.6209	0.5523	0.5637	0.5520	0.5487	0.5555
MP6	0.8260	0.8478	0.8481	0.8282	0.8291	0.7126	0.7184	0.7218	0.7217	0.7222	0.6039	0.6172	0.6178	0.6180	0.6201
y_1	0.8239	0.8461	0.8465	0.8262	0.8271	0.7098	0.7157	0.7192	0.7190	0.7196	0.6015	0.6148	0.6153	0.6156	0.6177
y_2	0.8250	0.8470	0.8474	0.8273	0.8281	0.7113	0.7171	0.7206	0.7204	0.7210	0.6027	0.6160	0.6166	0.6168	0.6189
y_3	0.8284	0.8497	0.8500	0.8305	0.8314	0.7159	0.7214	0.7249	0.7247	0.7253	0.6067	0.6201	0.6206	0.6209	0.6229
y_4	0.8266	0.8483	0.8486	0.8288	0.8297	0.7135	0.7192	0.7226	0.7225	0.7231	0.6046	0.6180	0.6185	0.6188	0.6208
MP5	0.8136	0.7811	0.8175	0.8176	0.8178	0.7411	0.7983	0.7299	0.7313	0.7410	0.6046	0.6163	0.6147	0.6161	0.6147
y_1	0.8150	0.7826	0.8188	0.8190	0.8191	0.7430	0.7996	0.7313	0.7327	0.7427	0.6059	0.6175	0.6158	0.6173	0.6158
y_2	0.8086	0.7761	0.8125	0.8127	0.8128	0.7352	0.7937	0.7248	0.7262	0.7354	0.6000	0.6117	0.6102	0.6117	0.6102
y_3	0.8175	0.7850	0.8214	0.8215	0.8216	0.7455	0.8017	0.7338	0.7352	0.7452	0.6082	0.6199	0.6182	0.6196	0.6182
y_4	0.8133	0.7808	0.8172	0.8174	0.8175	0.7406	0.7980	0.7297	0.7311	0.7406	0.6044	0.6162	0.6146	0.6160	0.6146
Water Quality	0.9762	0.9648	0.9743	0.9772	0.9862	0.9616	0.9497	0.9609	0.9674	0.9686	0.9464	0.9350	0.9474	0.9534	0.9545
y_1	0.9922	0.9797	0.9847	0.9878	0.9883	0.9772	0.9637	0.9688	0.9720	0.9688	0.9650	0.9510	0.9566	0.9609	0.9650
y_2	1.0181	1.0127	1.0161	1.0137	1.0064	1.0198	1.0109	1.0127	1.0153	1.0080	1.0073	0.9985	1.0076	1.0086	1.0075
y_3	0.9986	0.9805	0.9848	0.9849	0.9956	0.9918	0.9777	0.9849	0.9914	0.9864	0.9770	0.9708	0.9695	0.9794	0.9948
y_4	0.9636	0.9589	0.9731	0.9802	0.9816	0.9502	0.9445	0.9555	0.9640	0.9685	0.9255	0.9240	0.9290	0.9410	0.9461
y_5	0.9857	0.9779	0.9876	0.9893	0.9983	0.9641	0.9521	0.9705	0.9718	0.9799	0.9504	0.9429	0.9580	0.9581	0.9584
y_6	0.8971	0.8808	0.9109	0.9191	0.9432	0.8835	0.8746	0.8887	0.9136	0.9283	0.8659	0.8584	0.8687	0.8924	0.8779
y_7	1.0091	0.9957	1.0061	1.0072	1.0090	1.0009	0.9812	0.9997	1.0041	1.0081	0.9959	0.9739	0.9931	0.9990	0.9983
y_8	0.9681	0.9534	0.9497	0.9604	0.9761	0.9410	0.9357	0.9398	0.9490	0.9510	0.9409	0.9266	0.9312	0.9355	0.9469
y_9	0.9614	0.9383	0.9429	0.9448	0.9628	0.9411	0.9058	0.9357	0.9167	0.9103	0.8989	0.8791	0.9194	0.8867	0.9054
y_{10}	0.9939	0.9942	0.9994	0.9898	0.9952	0.9658	0.9625	0.9645	0.9829	0.9818	0.9432	0.9388	0.9418	0.9732	0.9501
y_{11}	0.9953	0.9778	0.9799	0.9874	0.9911	0.9820	0.9678	0.9825	0.9726	0.9734	0.9706	0.9497	0.9687	0.9632	0.9702
y_{12}	0.9708	0.9569	0.9748	0.9722	0.9904	0.9561	0.9547	0.9589	0.9674	0.9799	0.9298	0.9261	0.9402	0.9396	0.9588
y_{13}	1.0059	1.0063	1.0075	1.0102	1.0094	1.0051	0.9891	1.0047	1.0041	1.0047	1.0036	0.9891	1.0081	1.0031	1.0041
y_{14}	0.9069	0.8936	0.9233	0.9331	0.9593	0.8845	0.8757	0.8858	0.9193	0.9109	0.8751	0.8610	0.8713	0.9073	0.8794