MULTI-OUTPUT RANDOM FORESTS

Magisteruppsats i Informatik

Henrik Linusson

VT 2013:MAGI04

HÖGSKOLAN I BORÅS
INSTITUTIONEN HANDELS- OCH IT-HÖGSKOLAN

Svensk titel: Multi-Output Random Forests

Engelsk titel: Multi-Output Random Forests

Utgivningsår: 2013

Författare: Henrik Linusson

Handledare: Tuve Löfström

Abstract

The Random Forests ensemble predictor has proven to be well-suited for solving a multitude of different prediction problems. In this thesis, we propose an extension to the Random Forest framework that allows Random Forests to be constructed for multi-output decision problems with arbitrary combinations of classification and regression responses, with the goal of increasing predictive performance for such multi-output problems. We show that our method for combining decision tasks within the same decision tree reduces prediction error for most tasks compared to single-output decision trees based on the same node impurity metrics, and provide a comparison of different methods for combining such metrics.

Keywords: classification, multi-output, multi-task, Random Forest, regression

Sammanfattning

Ensemble-prediktorn Random Forest har visat sig vara väl anpassad för en mångfald av beslutsproblem. I denna uppsats föreslår vi en utökning till Random Forest-ramverket som tillåter konstruktion av Random Forest-skogar med stöd för problem med multipla responsvärden bestående av olika konstellationer av klassificerings- och regressionsproblem, med målet att minska prediktionsfelet för sådana problem. Vi visar att den föreslagna metoden för kombinering av beslutsproblem i ett gemensamt träd leder till minskat prediktionsfel för de flesta beslut jämfört med enkla träd baserade på samma beräkningar av orenhet i noder, och presenterar en jämförelse mellan olika metoder för att kombinera sådana beräkningar.

Nyckelord: klassificering, multi-output, multi-task, Random Forest, regression

Contents

List of Tables

Listings

1 Introduction

Multi-output pattern recognition problems are a special case of pattern recognition problems; whereas a typical pattern recognition problem involves a data set where each instance has a single (nominal or real-valued) output value, the instances in a multi-output pattern recognition problem have two or more output values — i.e., the output value is a vector rather than a scalar. There are two general approches for solving multi-output pattern recognition problems: either by transforming the problem into multiple single-output problems; or, by adapting a pattern recognition algorithm so that it directly handles multi-output data.

Training an inductive classifier or regression model can be a time consuming task — particularly so when training data sets are very large. When multiple models need to be trained using the same input data — but with different output data — time consumption can quickly get out of hand. Thus, for very large problems, the problem tranformation approach can prove to be unsuitable. Using the algorithm adaption approach, it is possible to directly create a model that simultaneously predicts a set of two or more classification labels, regression values, or even joint classification-regression outputs from only a single training iteration. More importantly, when the prediction tasks are related (i.e., there is a correlation or covariance between output values), traning a coherent multi-output model can potentially bring benefits in the form of increased predictive performance compared to training multiple disjoint models (Evgeniou and Pontil, 2004).

Ensemble classifiers and regression techniques, such as *bagging* (Breiman, 1996) and *boosting* (Freund and Schapire, 1996), are generelly regarded as de facto standard for constructing accurate prediction models. One often-successful implementation of bagging is the *Random Forest* predictor (Breiman, 2001) — Random Forest builds a collection of classification or regression trees from bootstrapped training data, and uses random feature subspacing to create diverse ensembles. Decision trees and forests have already been adapted to handle multiple classification outputs, multiple regression outputs and joint classification-regression outputs (Segal, 1992; Larsen and Speckman, 2004; Zhang, 1998; De'Ath, 2002; Faddoul et al., 2012; Segal and Xiao, 2011; Glocker et al., 2012) — however, none of these approaches handle data with arbitrary combinations of classification and regression outputs.

1.1 Problem Statement

Several approaches for solving multi-output prediction problems using decision trees and forests have been proposed, none of which are suited for solving decision problems consisting of arbitrary combinations of classification and regression tasks. It is shown that the Random Forest predictor provides accurate predictions for a multitude of different problems, thus we propose to develop a Random Forest extension that can successfully handle multi-output classification problems, multi-output regression problems, as well as arbitrary combina-

tions of classification and regression tasks.

1.2 Research Objectives

1. Design and implement a Random Forest predictor with support for multi-output problems consisting of arbitrary combinations of classification and regression tasks.

2. Evaluate the novel multi-output Random Forest predictor against alternative approaches for solving multi-output problems.

1.3 Main Contributions

In this paper, a novel multi-output Random Forest extension, inspired by Glocker et al. (2012), is proposed. This novel Random Forest supports multi-output problems with multiple classifiation outputs, multiple regression outputs, as well as arbitrary joint classification-regression outputs.

2 Acknowledgements

I would like to thank the following people for their help and encouragement throughout my work on this thesis:

- Tuve Löfström
- Ernst Ahlberg
- Lars Carlsson

3 Research Method

This study has been performed according to the criteria for design science research in informatics as proposed by Hevner et al. (2004). This chapter provides an overview of the research methods used to perform this study, as well as an overview of the study's adherence to the seven guidelines for design science research put forward by Hevner et al. (2004).

3.1 Research Design

This section presents an overview of the methods used to meet the two research objectives. Details on algorithm design and experimental design are presented in later chapters.

Research Objective 1: *Design and implement a Random Forest predictor with support multi-output problems consisting of arbitrary combinations of classification and regression tasks.*

To meet this first research objective, the Random Forest implementation developed by Linusson et al. (2012) has been modified to support the three different categories of multi-output problems (multi-output classification, multi-output regression and joint classification-regression). The design of of the Random Forest extension is based on related work proposed by Faddoul et al. (2012), Segal and Xiao (2011) and Glocker et al. (2012).

Research Objective 2: *Evaluate the novel multi-output Random Forest predictor against alternative approaches for solving multi-output problems.*

To meet the second research objective, the novel Random Forest predictor has been empirically evaluated on domain specific data, and results compared to alternative methods for solving multi-output decision problems.

Research Objective 1	Prescriptive artifact design and implementation
Research Objective 2	Experimental evaluation and comparison

The research presented in this thesis is deductive in nature — based on related literature, a hypthosesis is formulated in the form of a novel Random Forest that is expected to handle multi-output problems in a satisfactory manner. This hypothesis is then evaluated using a quantitative, experimental approach.

3.2 Design Science Research

The following section presents the seven guidelines for design science research in informatics proposed by Hevner et al. (2004), together with explainations of how this study aims to meet the guidelines.

Guideline 1: Design as an Artifact
The main goal of this study is to develop a Random Forest implementation that handles different types of multi-output prediction problems — specifically, the RF implementation developed by Linusson et al. (2012) is modified to provide support for multi-output classification problems, multivariate regression problems and joint classification-regression problems. As such, the resulting RF implementation serves as an artifact that illustrates the utility of proposed algorithm adaptions, as well as a functional tool for multiple-output prediction.

Guideline 2: Problem Relevance
The motivating application for this RF implementation reimans *in silico* experiments on QSAR models as discussed by Linusson et al. (2012). Typically a multitude of activities are predicted for the chemical compounds of drug candidates, and predicting these activities in one coherent model rather than several individual models provides a possibility for increased predictive performance and time efficiency.

Guideline 3: Design Evaluation
The multi-output Random Forest implementaion is evaluated empirically against single-output Random Forests using domain specific bioinformatics data (Gilman et al., 1990; Krejsa et al., 2003).

Guideline 4: Research Contributions
As part of the design of the multi-output Random Forest implementation, a new method for combining classification and regression tasks within a single decision forest model is proposed. The proposed method is empirically evaluated against other methods for solving multi-output decision problems; in addition, several different methods for aggregating node impurity measures are empirically evaluated in the context of the novel combination method.

Guideline 5: Research Rigor
The empirical evaluation of the novel Random Forest implementation follows a scientific experimental design, that has been thoroughly explained to allow for critical review.

Guideline 6: Design as a Search Process
Design of the multi-output Random Forest implementation is grounded in published scientific literature, outlined in chapter 4.

Guideline 7: Communication of Research
Relevant design choices and algorithm modifications are thoroughly described in this report.

4 Theoretical Framework

In this chapter we present existing theory on Random Forests, multi-output prediction problems, as well as multi-output decision trees.

4.1 C4.5

Proposed by Quinlan (1993), C4.5 is a decision tree induction algorithm that utilizes *entropy* and *information gain* to recursively divide data into increasingly pure partitions. Given a data set of examples $zs = \{(x_1, y_1), (x_2, y_2), ...(x_n, y_n)\}$, the goal is to train a decision tree such that its leaf nodes contain instances with as low variance as possible with regards to output values — this is accomplished by, in each node, splitting the examples into two partitions based on the feature-value pair that best separates the data, until no gain can be had from further partitioning.

Finding the feature-value pair that best separates the data in node t is done by an exhaustive search over all combinations appearing within the node — for each feature f in the data set, and each value $f(x_i)$ in node t, a split is attempted and the purity of the resulting child nodes is calculated. The examples in t are then partitioned according to the feature-value pair (f, x) that provides the highest increase in node purity in the child nodes.

The relative purity of a split is calculated using information gain — the difference in entropy between the outputs of all examples in node t and the outputs of the examples in t partitioned according to a feature-value pair (f, x). Information gain is calculated as

$$IG(t; f, x) = H(t) - H(t|f, x), \tag{1}$$

where $H(t)$ is the Shannon entropy of node t. In turn, Shannon entropy is calculated as

$$H(t) = -\sum_{y \in Y} p(y|t) \log p(y|t), \tag{2}$$

where $p(y|t)$ is the relative frequency of class y in node t. As splitting a node into k partitions results in each of the child nodes containing fewer instances than the parent node, it is necessary to scale the relative entropy according to partition size — thus, the information gain from dividing a node into k partitions is calculated as

$$IG(t; f, x) = H(t) - \sum_{i=1}^{k} \frac{n_i}{n_p} H(t_i|f, x), \tag{3}$$

where $H(t)$ is the Shannon entropy (2) of the parent node, the node is split into k partitions (two for a binary tree), n_p and n_i denote the number of examples in the parent node and each child node respectively, and $H(t_i|f, x)$ is the Shannon entropy of a given child node t_i.

When no gain can be had from further partitioning the data (e.g. node t contains only examples of a single class), t is made to be a leaf in the tree, and is marked with the majority class amongst examples within the node. When predicting output values for a novel instance x_j using a C4.5 decision tree, the feature values of x_j is tested according to the feature-value pairs stored in each node, and the instance is sent to appropriate subtree. This process continues recursively until a leaf node is encountered, and the leaf's stored majority class is the prediction output for x_j.

4.2 Random Forest

The Random Forest predictor (Breiman, 2001) is an ensemble predictor that combines *bagging* (Breiman, 1996) and *random feature subspacing* (Ho, 1998) to construct accurate classification or regression ensembles of decision trees. Both bagging and random subspacing are effective ensemble methods in themselves — both implicitly increase diversity amongst classifiers by limiting their scope, and effectively forcing them to learn a rather specialized prediction rule. During prediction, each of the individual ensemble members 'vote' for the prediction that best resembles its limited scope — these votes are then combined using an aggregation method to compute the final ensemble prediction.

4.2.1 Bagging

Bagging (bootstrap aggregation) (Breiman, 1996) introduces diversity in the predictor ensemble by training each ensemble member on a separate bootstrap sample of the training data — each bootstrap sample is constructed by drawing (with replacement) N examples from the training data, where N is the number of examples in the original training data; thus, the resulting bootstrap data contains the same amount of instances as the original data set, but some of these (approximately 1/3) are duplicates, and approximatly 1/3 of the instances are left out of the bootstrap sample.

Bagging has proven to be an effective method for constructing classification and regression ensembles — however, it is not without its limitations. Most notably, bagged ensembles work well only with *unstable* classifiers — e.g. classification and regression trees (CART) — classifiers whose inferred rules are significantly affected by changes in the training data (Breiman, 1996).

4.2.2 Random Feature Subspacing

Considering the training data set to be a two-dimensional $N \times M$ matrix — where features are represented by columns, example instances are represented by rows, and a single cell represents a particular instance's value for a certain feature — it is possible to imagine bootstrap sampling simply as repeatedly selecting (with replacement) a random row from the matrix and adding it to the current bootstrap sample, until the size of the bootstrap sample equals that of the original matrix.

Table 1: Bagging vs. Random Subspace Method

(a) Bagging

	f_1	f_2	f_3	f_4	f_5
x_1	1	2	3	5	2
x_2	5	1	2	1	3
x_3	4	2	1	2	5
x_4	3	2	1	5	6
x_5	1	2	3	1	2

(b) Random Subspace Method

	f_1	f_2	f_3	f_4	f_5
x_1	1	2	3	5	2
x_2	5	1	2	1	3
x_3	4	2	1	2	5
x_4	3	2	1	5	6
x_5	1	2	3	1	2

Table 2: Random Forest (Bagging + Random Subspace Method)

	f_1	f_2	f_3	f_4	f_5
x_1	1	2	3	5	2
x_2	5	1	2	1	3
x_3	4	2	1	2	5
x_4	3	2	1	5	6
x_5	1	2	3	1	2

The random subspace method (Ho, 1998) operates in a similar manner, with three key differences: (1) instead of randomly selecting rows (instances), columns (features) are randomly included in the training data; (2) sampling is done without replacement; and (3) rather than selecting M columns, a subset of size $m < M$ is selected. A training sample created using the random subspace method thus contains all the original N example instances, each with the same randomly reduced feature space.

Random Forest combines the two sampling methods, which results in a two-dimensional randomized data reduction — thus, each tree in the forest is forced to focus on a limited problem space. This increases the diversity amongst the ensemble members, resulting in accurate decision forests.

4.2.3 Classifier Induction

As the Random Forest classifier is an ensemble classifier, its induction is best described from a hierarchial perspective, considering first the the induction of the entire ensemble while abstracting the induction of the individual trees, and second consider the induction of an individual tree while abstracting the ensemble meta process.

Inspection of the induction of a Random Forest classifier from a bird's-eye perspective reveals a very simple iterative two-step process: (1) draw a bootstrap sample B'_i from the training data B, and (2) train a tree model using B'_i as training data; repeat (1) and (2) until n_{tree} trees have been trained. Hence, at this ensemble-wide level, the bagging procedure is applied to ensure diversity amongst classifiers — however, to spot the application of random feature subspacing, it is necessary to look more closely at the actual tree induction

algorithm.

The CART trees used in Random Forest much resemble C4.5 trees, but have a couple of key differences. As proposed by Breiman (2001), RF-CART uses *Gini-index* instead of information gain to compute node purity; however, an argument can be made that this is a design choice pertaining to CART rather than Random Forest — RF could very well be made to work with trees that use information gain as splitting criteria. The most interesting difference between the two trees is the application of random feature subspacing in RF-CART trees — whereas C4.5 examines all possible features to find the best split in any given node, the trees in Random Forest examine only a random subsample of features in each node.

4.2.4 Classifier Combination

Predicting output values for novel instances with a Random Forest predictor involves letting each individual ensemble member vote for the most probable output according to its learned decision rule. The ensemble members' votes are tallied and aggregated — typically using mode for classification and mean for regression — into a common ensemble output.

4.3 Multi-Output Classification

Most often, classification problems involve *single-label* classification — that is, the goal is to learn a classification rule whose output is a single label $\hat{y}$ from a set of disjoint labels Y. In *multi-output* classification however, the problem is slightly different — the goal here is to learn a classification rule whose output is a set, or vector, of labels $\hat{v} = \{\hat{y}_1 \in Y_1, \hat{y}_2 \in Y_2, ..., \hat{y}_n \in Y_n\}$.

Note here that multi-output classification differs from the very closely related problem of *multi-label* classification. Both problems involve classification with multiple outputs, but differ in application and assumptions made regarding the size of the output. Multi-label classification involves classifying instances into several labels that share semantics (Tsoumakas and Katakis, 2007); consider for example the problem of classifying songs according to their genre — it is possible to classify a song as either **pop** or **rock**, but also possible to classify a song as both (i.e. "pop rock"). Here, **pop** and **rock** share semantics: they both relate to the songs' genre, and are thus two different values of the same label. There is also no *a priori* knowledge regarding the size of the output — it is very possible that a song cannot be classified as any previously known genre, or that it is best classified as several different genres.

The problem of multi-output classification is effectively the opposite of multi-label classification — the output values do not share semantics, but the number of outputs is known *a priori*. Consider a classification problem where the goal is to simultaneously predict temperature (**low**, **medium** or **high**) and pressure (**low**, **medium** or **high**) inside a pressure cooker — in this case the model is expected to output exactly two values, one value for the temperature label and another for the pressure label. While the two labels share the same set

of possible values, their semantics differ as temperature and pressure measure two different concepts. The machine learning task of solving a multi-output problem thus involves building a predictive model that simultaneously outputs a set of (two or more) labels that measure different concepts — essentially two or more separate (although related) classification problems are solved concurrently within the same model. Another term for multi-output classification is *multi-task* classification — a term that perhaps better illustrates the fact that a multi-output classification problem is effectively equivalent to multiple simultaneous (multi-tasked) single-label classification problems.

An observant reader might quickly realize that it is possible to transform a multi-label classification problem into a multi-output classification problem — the example problem of classifying songs according to their genre is easily transformed to a multi-output binary problem by letting each value of the genre label constitute a separate binary label. Thus, **pop** becomes a binary label with values *true* and *false*, and every song is classified as either **pop**=*true* or **pop**=*false*; similarly, every song is classified as either **rock**=*true* or **rock**=*false*. In contrast to the multi-label version of the problem, where each label is either included in or excluded from the set of outputs, the multi-output version of the problem explicitly classifies every song as either belonging to a specific genre or not, and the model is expected to output exactly one value (*true* or *false*) per genre. Note also that, while multi-label problems can be transformed to multi-output problems, the opposite is not necessarily true.

Tsoumakas and Katakis (2007) describe and categorize different approaches for solving multi-label classification problems; as the focus of this paper lies solely on multi-output problems, many of these approaches are not applicable — however, their categorization still holds interest for multi-output problems. The two different categories of solutions to multi-label problems proposed by Tsoumakas and Katakis (2007) are: (1) problem transformation methods, and (2) algorithm adaption methods. Problem transformation involves modifying the problem so that it can be solved using standard classification methods, e.g. by transforming a multi-label (or multi-output) problem into multiple single-output problems. Algorithm adaption methods instead aim to modify existing classification methods so that they can handle multi-label (or multi-output) data directly.

4.3.1 Problem Transformation Methods

Tsoumakas and Katakis (2007) summarize five different approaches for transforming multi-label problems into single-label problems — of these, only one is of particular interest for multi-output problems.

As illustrated in the previous section, a multi-label classification problem can be transformed into a multi-output classification problem simply by letting each value constitute a separate binary label. Similarly, a problem transformation approach can be applied to multi-output classification problems to simplify their solution.

We already noted that multi-output classification is effectively equivalent to

multi-tasking several separate (but related) single-label classification problems — a naïve solution to a multi-output classification problem is thus to simply disregard the multi-tasking part of the problem. Instead of training a classifier that simultaneously predicts $|\Upsilon|$ outputs, it is possible to train $|\Upsilon|$ separate classifiers — one for each label $Y \in \Upsilon$. However, attempting to solve a multi-output classification problem in this manner should not be done without consideration — training $|\Upsilon|$ separate classifiers might prove to be intractable, or at the very least impractical, when $|\Upsilon|$ is very large. In such cases, it might prove more practical to select an algorithm adaption method that allows for the training of a single model with $|\Upsilon|$ outputs.

4.3.2 Algorithm Adaption Methods

The simplicity of the problem transformation approach renders it quite suitable for problems where its downsides have little or no effect — for larger problems however the algorithm adaption approach might prove more effective. Additionally, empirical evidence suggests that learning related tasks simultaneously rather than independently can improve predictive performance (Evgeniou and Pontil, 2004). On the other hand, if the tasks are very dissimilar, predictive performance might suffer when the tasks are learned together rather than independently (Faddoul et al., 2010). Thus, we are able to conclude the following: if the tasks we want our predictor to learn are related, we should aim to find a suitable algorithm adaption method; if the tasks we want to learn are unrelated, we should instead aim to find a suitable problem transformation method. Finally, we should consider the problem size, and realize that when the tasks are unrelated there is a potential trade-off between time efficiency and predictive performance in selecting either a problem transformation method or an algorithm adaption method. Problem transformation methods may see increased predictive performance for unrelated tasks, but suffer in time efficiency for large problems, and vice versa.

4.3.3 Multi-Task Decision-Trees

Faddoul et al. (2012) propose a modified version of the C4.5 decision tree learning algorithm (Quinlan, 1993), that directly handles multi-output classification problems. The modified version (dubbed `MT-DT`) differs from the standard C4.5 implementation in two critical aspects: its node splitting criteria and its decision process (Faddoul et al., 2012).

Faddoul et al. (2012) propose three different approaches for combining multiple single-task information gain measures into a single multi-task information gain measure: *joint information gain*, *unweighted sum*, and *maximum information gain*. The joint information gain is defined using a concatenation of all the individual tasks, i.e., the relative difference in entropy as measured over all decision tasks. Faddoul et al. (2012) show that the unweighted sum (4) of individual information gains of all the tasks is equivalent to the joint information gain.

$$IG_U(t; f, x) = \sum_{Y \in \Upsilon} IG_Y(t; f, x). \tag{4}$$

Max information gain, as proposed by Faddoul et al. (2012), is defined simply as the maximum information gain amongst all tasks:

$$IG_M(t; f, x) = max\left\{IG_Y(t; f, x), Y \in \Upsilon\right\} \tag{5}$$

Empirical results provided by Faddoul et al. (2012), where `MT-DT` trees make out the ensemble members of a boosted decision forest, suggest that IG_M provides superior predictive accuracy compared to both IG_J and IG_U.

Multi-Task Tree Induction

In a single-label classification setting, a decision tree induction algorithm (such as C4.5) recursively splits nodes, adding (typically two) children until it is possible to create a leaf such that a large majority (or even all) of its example instances belong to the same class. In a multi-output setting however, tree induction is not necessarily quite as simple. Consider a multi-output classification problem with two binary outputs υ_1 and υ_2; it is possible that after t splits, a node contains only positive values for υ_1, but a mixture of positive and negative values for υ_2 — hence, when constructing decision trees for multiple simultaneous tasks, it is necessary to keep in mind that the decision process for a certain task might require a shorter decision path than other tasks within the same multi-output problem (Faddoul et al., 2012). `MT-DT` handles this by checking, in every node, whether it is possible to create a stop-node for any of the tasks — for the example above, this would result in a tree where an internal node t_1 is marked as a stop node for υ_1, labeled with the positive class. Since the goal is to make predictions for both binary outputs, t_1 is not made to be a leaf node — instead, recursive splitting continues from t_1 until a node t_2 is found such that t_2 is pure enough with regards to υ_2 so that a classification rule can be made for the second binary task. At this point, decision nodes (internal or leaf nodes) have been found for all outputs (υ_1 and υ_2) and the recursive tree induction algorithm can terminate.

Multi-Task Tree Classification

Unsurprisingly, classification using an already-built `MT-DT` model follows the same formula as its induction — during traversal of the tree, every node is checked to determine whether a decision can be made for any of the currently undecided tasks. In the υ_1, υ_2 example, a classification would be made for υ_1 in node t_1, as it is marked as a stop-node for υ_1; traversal then continues until t_2 is encountered, and a classification can be made for υ_2. At this point, all outputs have been classified, and the traversal can terminate, returning the two values at t_1 and t_2 as the classifications for υ_1 and υ_2 respectively.

4.4 Multivariate Regression

While regular regression problems have a single real-valued output measuring a single concept, multivariate regression problems have $|\Upsilon|$ different outputs, each measuring one of $|\Upsilon|$ different concepts.

As with the case of multi-output classification, there are two distinct approaches for attempting to solve a multivariate regression problem: either transform the problem so that it can be solved using regular single-output regressors, or adapt the algorithm into directly handling multiple outputs. The naïve approach of transforming a multi-output problem into several single-output problems is applicable for regression just as with classification, and the same caveats apply.

4.4.1 Multivariate Regression Trees

Segal (1992) proposes a design for multivariate regression trees (`MRT`) that are able to make predictions for multiple related regression tasks; these multivariate regression trees are based on the least squares split function proposed in the CART framweork (Breiman et al., 1984). For a unnivariate-response regression tree, the least squares function aims to minimize the sum of squared errors amongst the child nodes. Thus, the task is to minimize:

$$\phi(s,t) = SS(t) - SS(t_L) - SS(t_R), \tag{6}$$

where SS(t) is the sum of squared errors in node t, defined as

$$SS(t) = \sum_{i \in t} (y_i - \bar{y}(t))^2. \tag{7}$$

Segal (1992) adds a covariance weighting to the squared error, which drives the tree induction algorithm into forming child nodes that represent homogenous clusters with regard to the set of output responses:

$$SS(t) = \sum_{i \in t} (y_i - \bar{y}(t))' V^{-1}(t,\eta)(y_i - \bar{y}(t)). \tag{8}$$

$V(t)$ here denotes the covariance matrix of t, and η represents parameters for this covariance structure. Segal and Xiao (2011) propose an extension to the Random Forest ensemble predictor that handles multivariate responses — extension is done simply by replacing the typical univariate trees in the Random Forest with the same type of multivariate trees as those proposed by Segal (1992).

4.5 Joint Classification-Regression Problems

As previously discussed, one key motivation for attempting to solve multi-output pattern recognition problems using algorithm adaption methods is the expectation that a single model trained on a set of related tasks will show an improvement in predictive performance as compared to a set of individual models

each trained on a single task. This raises the question: what if the multi-output problem contains both classification tasks and regression tasks?

If the tasks are unrelated, solving such a jont classification-regression problem needs not be more difficult than training a set of classifiers and regressors on the individual tasks; however, if the tasks are related, we expect an algorithm adaption method to provide the best results in terms of predictive performance.

4.5.1 Joint Classification-Regression Trees

Glocker et al. (2012) propose a tree induction algorithm that simultaneously solves one classification task and one regression task. Much like `MT-DT` and `MRT`, the joint classification-regression tree (`JCRT`) solves multiple simultaneous prediction tasks by modifying the node-split function in the inductive step, and marking terminal nodes with appropriate values for each task.

Due to the nature of joint classification-regression problems, the modified split function is required to consider the error of both the classification part and the regression part simultaneously. The split function proposed by Glocker et al. (2012) uses an entropy function consisting of three parts: first of all, Shannon entropy is computed for the classification part; second, a weighted differential entropy is calculated for the regression part; third, due to the fact that Shannon entropies and differential entropies exist in different ranges, a normalization step is applied to combine the two entropies. The Shannon entropy is calculated just as previously described:

$$H_c(t) = -\sum_{c \in C} p(c|x) \log p(c|x). \tag{9}$$

The differential entropy measure used by Glocker et al. (2012) for the regression part of the problem is calculated in a similar manner, with two key differences: rather than summing probabilities of nominal values, the entropy is defined by the differential of the probability function of the real-valued output; additionally, the probability function is weighted on a per-class basis:

$$H_{r|c}(t) = \sum_{c \in C} p(c|x) \left(-\int_{r \in \mathbb{R}^n} p(r|c,x) \log p(r|c,x)\, dr \right). \tag{10}$$

Normalization is performed for both tasks with regard to the entropies in the root node:

$$H(t) = \frac{1}{2} \left(\frac{H_c(t)}{H_c(t_0)} + \frac{H_{r|c}(t)}{H_{r|c}(t_0)} \right). \tag{11}$$

Glocker et al. (2012) use `JCRT` to construct joint classification-regression forests, which they evaluate on spatially structured data in the form of CT scans where the two tasks are: (1) classify pixels according to objects (organs), and (2) estimate the distance from each pixel to the object's boundary. Glocker et al. (2012) show that these joint forests are not only able to provide accurate estimates of object boundaries, but also improve the accuracy of the classification of objects.

4.6 Summary

Segal (1992) introduced multivariate regression trees as a means for improving predictive performance in the analysis of longitudinal data. The multivariate regression trees proposed by Segal extends the CART framework introduced by Breiman et al. (1984) by modifying the split functions used during tree induction. By considering each of the responses and their covariance during node splitting, the multivariate regression trees are better able to make accurate predictions when the responses are covariant. De'Ath (2002) applies a similar strategy to produce multivariate regression trees aimed at solving decision problems for geographical and ecological data, using distance based splitting criteria to minimize dissimilarities between geographical clusters. Segal and Xiao (2011) propose a multivariate Random Forest ensemble (Breiman, 2001) using the covariance weighted multivariate regression trees proposed by Segal (1992), that show an increased predictive performance compared to other methods for multivariate prediction, as well as univariate prediction.

Zhang (1998) applies an approach similar to that proposed by Segal (1992) on binary classification problems, resulting in multi-output classification trees using splits based on covariance weighted entropy. Faddoul et al. (2010) introduce a Boosting ensemble (Freund and Schapire, 1996) based on stumps of multi-output classification trees, and propose simplified methods for calculating node splits for multiple classification tasks (Faddoul et al., 2012).

Glocker et al. (2012) propose a method for learning two related tasks with different response types (one classification task and one regression task) within the same decision tree, and show that both tasks are more accurately predicted together than separately using forests of joint classification-regression trees.

5 Novel Multi-Output Random Forest

To provide a general joint classification-regression Random Forest implementation, we propose that the split function used by Glocker et al. (2012) be generalized such that it handles an arbitrary amount of classficiation and regression tasks within the same data set. This chapter provides a description of our novel Random Forest implementation based on such a split function.

5.1 Aggregated Information Gain

For problems with a single classification output and a single regression output, as handled by `JCRT`, the entropy of a node is calculated using the probability $p(c, r|x)$ which, using the chain rule, equates to $p(c|x)p(r|c, x)$, which in turn can be computed using the joint entropy function proposed by Glocker et al. (2012). For a general problem with an arbitrary amount of outputs we are instead required to calculate the entropy of a node using the probability $p(c_1, ..., c_n, r_1, ..., r_n|x)$ and applying the chain rule to this probability would result in a combinatorial mess. To avoid this, we instead propose a split function that evaluates each task independently. Thus, we calculate the impurity of each classification task using the standard Shannon entropy equation:

$$H_{c,i}(t) = -\sum_{c \in C_i} p(c|x) \log p(c|x), \tag{12}$$

and for regression tasks, we calculate the (unweighted) differential entropy:

$$H_{r,j}(t) = -\int_{r \in \mathbb{R}_j^n} p(r|x) \log p(r|x)\, dr. \tag{13}$$

Normalization of the entropy for each individual task can then be performed with regards to the root node as such:

$$H_{norm,i}(t) = \frac{H_i(t)}{H_i(t_0)}. \tag{14}$$

From this point, we can calculate the normalized joint entropy as the average normalized entropy of the individual tasks:

$$H(t) = \frac{1}{N} \sum_{i=1}^{N} H_{norm,i}(t), \tag{15}$$

and then calculate the normalized joint information gain using (1). However, as we are considering each task individually, it is possible to not only calculate their individual entropies, but also their individual information gains:

$$IG_i(t; f, x) = H_{norm,i}(t) - H_{norm,i}(t|f, x). \tag{16}$$

This gives us the option to aggregate the information gains using either the unweighted sum of information gains (4) or the maximum information gain (5).

5.2 Random Gain

Borrowing from the ideas behind the Random Forest ensemble predictor (Breiman, 2001), we propose an additional method for aggregating the normalized information gains of the individual tasks: *random information gain*. Here, we randomly select one task, and output its individual information gain as the information gain for the split:

$$IG_R(t; f, x) = random\left\{IG_Y(t; f, x), Y \in \Upsilon\right\}. \tag{17}$$

This random information gain selection effectively adds another level of diversity to the decision trees, as they are (during each split) steered towards focusing on different tasks. As diversity is an important component in successful ensemble predictors, random information gain selection is expected to provide good performance despite its unpredictable nature.

Note that we apply random information gain in the same manner as max information gain is applied — for each split attempt within a node, the information gain of a (possibly different) random task is selected as the split's information gain. It would of course be possible to first select a random task, and then split the node on the feature-value pair that maximizes the information gain of this common task, however, such an approach is not evaluated in this thesis.

5.3 Feature Selection

As this Random Forest implementation is intended for sparse data structures, the sparse feature selection method proposed by Linusson et al. (2012) is used — when selecting a set of random features on which to attempt node partitioning, we make sure that the selected features are represented by at least one non-null value within the node.

5.4 Split Value Selection

During our empirical evaluation, we employ two different methods for selecting split values during node partitioning:

1. A search is performed over a subset of equally spaced feature values, as performed by Linusson et al. (2012).

2. As suggested by Boström (2011), rather than searching for a split value, two random feature values are selected, and a split is attempted at their mean. Unlike Boström (2011), we make no considerations regarding output values when selecting random feature values in the multi-output case, as we are less likely to select two feature values for which none of the tasks contain opposite class labels or dissimilar regression values.

6 Empirical Evaluation

This chapter describes the empirical evaluation of the multi-output Random Forest, performed in comparison to two single-output Random Forests. We present the data sets and settings used for evaluation, then proceed to present our results and analysis.

6.1 Data Sets

Evaluation of the multiple-output Random Forest implementation has been performed on two bioinformatics data sets. The first data set, courtesy of Gilman et al. (1990) contains 8 different pharmacokinetical activities of 320 chemical compounds.

Table 3: Pharmacokinetic Responses (Gilman et al., 1990)

Response Name	**Value Type**
LP — LIPO	LogP
OA — Oral Availability	%
UE — Urinary Excretion	%
PB — Plasma Bound	%
Clr — Clearance	ml/(min*kg)
VoD — Volume of Distribution	L/kg
HL — Half Life	Hours
D — Distribution	LogD

The second dataset, courtesy of Krejsa et al. (2003), contains 200 QSAR activities (mainly inhibition percentages) of 2401 chemical compounds. 10 of these activities were randomly selected to evaluate our Random Forest predictor.

Table 4: Data Set Metadata

Dataset	**Instances**	**Features**	**Output Values**
Gilman et al. (1990)	320	5680	8
Krejsa et al. (2003)	2401	22734	10 (200)

Both data sets contain only regression tasks — however, the second data set (Krejsa et al., 2003) is associated with a set of assay specific cutoffs that allow us to transform the regression tasks into binary classification tasks.

Table 5: Assay Ranges and Classification Cutoffs Krejsa et al. (2003)

Assay ID	Min	Max	Cutoff	Positive Instances
100006	−47.00%	102.00%	> 30%	328
100029	−226.00%	97.00%	> 30%	486
100045	−78.00%	271.00%	< 60%	449
100077	−49.00%	100.00%	> 30%	12
100087	30.00%	100.00%	N/A	N/A
100115	−47.00%	108.00%	> 30%	239
100135	−103.00%	1140.00%	> 30%	380
100160	−49.00%	100.00%	> 30%	55
100220	−49.00%	110.00%	> 30%	90
100221	−80.00%	100.00%	> 30%	181

To allow us to perform classification for the first data set, we divide the tasks based on their median value.

Table 6: Median Cutoffs Gilman et al. (1990)

Response Name	Median Cutoff	Positive Instances
LIPO	< 1.94	152
Oral Availability	< 62.00	139
Urinary Excretion	< 15.00	156
Plasma Bound	< 72.00	148
Clearance	< 4.00	158
Volume of Distribution	< 1.00	158
Half Life	< 4.00	163
Distribution	< 0.76	152

Missing Output Values
Unlike single-output problems, where all training instances are typically associated with an output value (otherwise they are, under normal circumstances, quite useless as training examples), training instances in a multi-output data set can potentially be associated with output values for only a subset of all tasks. For the purpose of this study, missing output values in the data sets have been replaced with the median value for the respective task both in training and test data, hence, reported error values are likely overly optimistic. However, this method is used for all tested Random Forest methods (including single-output variants), and thus a comparative analysis should remain fair.

6.2 Random Forests

The novel multi-output Random Forest implementation is evaluated against two single-output Random Forests — `SPARF` (Linusson et al., 2012), based on Gini index for classification and variance for regression, and a modified version of

`SPARF` that is based on information gain for both classification and regression. We use the following shorthands to refer to the three different Random Forest implementations:

SPARF	SPArse Random Forest (Linusson et al., 2012) Gini index/variance
SRF	Single-output RF, information gain (entropy/differential entropy)
MRF	Multi-output RF, information gain (entropy/differential entropy)

Additionally, we use the following shorthands to refer to the different information gain aggregation methods:

-R-	Random information gain
-S-	Unweighted sum information gain
-M-	Max information gain
-J-	Joint information gain
-N-	Normalized entropy

Lastly, we use the following shorthand to refer to random split-value selection (Boström, 2011):

-RV	Random value split

Hence, by `SRF-RV` we refer to single-output Random Forest using information gain (based on entropy for classification tasks and differential entropy for regression tasks) and random value selection during node splitting; and, by `MRF-N-R-RV`, we refer to multi-output Random Forest using normalized random information gain and random value selection.

6.3 Experiment Setup

The two datasets, GG (Gilman et al., 1990) and BP (Krejsa et al., 2003), were run using different mixes of classification and regression tasks:

Run	**Classification Tasks**	**Regression Tasks**
GG-FullR	-	All
GG-FullC	All	-
GG-Joint	1-4	5-8
GG-Joint2	8	1-7
GG-Joint3	1	2-8
BP-FullR	-	All
BP-Joint	1-4, 6-10	5

Experiments were first run on the full-regression and full-classification versions of the GG data set — candidates showing the most promising results were then selected and run on the joint GG data sets, as well as the larger BP data set. All results presented are based on 10×10-fold cross-validation, using the following common settings: 500 trees, $\log_2 M$ features.

6.4 Analysis Method

For each task, we present the predictive performance of all tested Random Forests — for classification tasks we present their *accuracy*; and for regression tasks, we present their *root-mean-square error* (RMSE). The Random Forests are ranked based on their performance on each task; the best performer is assigned rank 1, the second best performer is assigned rank 2, etc. Tied error scores result in an averaged rank (i.e., if two predictors are tied for first place, they are both assigned rank 1.5). Based on these rankings, we also calculate the average rank of each predictor. These average ranks serve two purposes:

1. They allow us to easily discern which Random Forests provide the best predictive performance on average.
2. They allow us to apply the Nemenyi post-hoc test (Nemenyi, 1963) to determine whether differences in predictive performance are statistically significant.

Accuracy
The accuracy of a classifier is defined as the fraction of correct classifications made on the test data set.

$$ACC(h) = \frac{|\{(x,y) \in Z_{test} \mid h(x) = y\}|}{|Z_{test}|} \tag{18}$$

RMSE
Just as the name suggets, the root-mean-square error of a regression predictor is the square-root of the sum of squared errors (the difference between the prediction and the correct output) as measured on the test data set.

$$RMSE(h) = \sqrt{\frac{\sum_{(x,y)\in Z_{test}} (y - h(x))^2}{|Z_{test}|}} \tag{19}$$

7 Results

Here we present the results from the different runs outlined in the Experiment Setup section.

7.1 GG-FullR and GG-FullC

For the GG-FullR and GG-FullC data sets, all split functions (sum gain, max gain, random gain, and joint gain) were evaluated using both random value selection (Boström, 2011) and a search over a reduced value space (Linusson et al., 2012). Sum gain, max gain and random gain were run both with and without normalization, while joint gain was run only using normalization.

Tables 7 and 8 show the RMSE and accuracy of each predictor on the GG-FullR and GG-FullC data sets respectively.

Table 7: GG-FullR RMSE

GG-FullR	L	OA	UE	PB	Clr	VoD	HL	D
SPARF	1.41	26.51	24.27	24.48	50.95	13.05	43.76	1.71
SRF	1.44	26.30	24.45	25.53	49.74	12.85	41.84	1.71
MRF-S	1.41	26.38	24.54	25.69	48.58	12.51	39.78	1.70
MRF-M	1.37	26.13	24.19	25.27	50.15	12.58	40.16	1.65
MRF-R	1.37	26.02	24.10	25.20	50.23	12.55	39.63	1.65
MRF-S-RV	1.43	26.40	24.36	25.83	47.99	12.54	39.53	1.70
MRF-M-RV	1.40	26.01	24.01	25.31	49.32	12.34	38.69	1.66
MRF-R-RV	1.40	25.96	23.98	25.42	49.27	12.52	39.62	1.66
MRF-N-S	1.41	26.48	24.48	25.72	48.07	12.60	39.25	1.70
MRF-N-M	1.37	26.04	24.13	25.26	49.98	12.57	40.28	1.65
MRF-N-R	1.37	26.11	24.12	25.11	49.93	12.45	38.78	1.65
MRF-N-J	1.41	26.42	24.51	25.74	48.40	12.56	39.65	1.70
MRF-N-S-RV	1.44	26.34	24.38	25.87	47.93	12.49	38.48	1.70
MRF-N-M-RV	1.40	26.03	24.07	25.33	48.75	12.54	38.85	1.65
MRF-N-R-RV	1.40	26.04	23.91	25.33	49.26	12.47	38.82	1.65
MRF-N-J-RV	1.43	26.33	24.33	25.81	48.28	12.52	38.72	1.70

Table 8: GG-FullC Accuracy%

GG-FullC	L	OA	UE	PB	Clr	VoD	HL	D
SPARF	84.16	67.06	72.91	76.84	71.81	77.53	66.69	79.47
SRF	83.75	68.78	68.53	72.88	70.31	75.41	67.53	78.94
MRF-S	83.31	67.91	67.22	73.25	69.66	76.28	68.19	78.16
MRF-M	82.16	67.47	70.50	76.91	70.44	76.06	69.78	78.19
MRF-R	82.59	66.09	69.56	75.78	69.53	76.75	70.28	77.34
MRF-S-RV	83.34	67.91	68.03	73.06	70.69	76.94	67.97	78.00
MRF-M-RV	83.28	67.81	70.38	75.63	71.16	77.03	69.50	78.31
MRF-R-RV	83.81	67.03	68.63	74.19	69.19	77.50	68.97	79.22
MRF-N-S	83.19	67.78	67.50	73.69	69.47	77.06	67.63	77.88
MRF-N-M	82.38	67.97	71.59	76.88	69.78	70.47	76.50	78.22
MRF-N-R	83.06	66.72	70.25	75.84	70.31	76.94	70.88	77.81
MRF-N-J	83.41	67.75	67.75	73.50	69.66	76.84	68.19	77.47
MRF-N-S-RV	83.88	67.59	68.22	73.13	69.59	76.97	68.66	79.50
MRF-N-M-RV	83.84	67.91	70.91	76.03	71.28	77.41	70.25	78.38
MRF-N-R-RV	83.97	66.50	69.53	74.69	70.38	78.13	69.38	79.28
MRF-N-J-RV	78.25	67.09	68.09	73.06	70.34	83.69	77.38	67.94

Table 9 and 10 show the per-task ranks and average ranks for each of the Random Forest predictors.

Table 9: GG-FullR Rankings

GG-FullR	L	OA	UE	PB	Clr	VoD	HL	D	Avg
SPARF	10.5	16	9	1	16	16	16	15.5	12.5
SRF	15.5	9	13	10	11	15	15	15.5	13
MRF-S	10.5	12	16	11	6	5	12	13.8	10.8
MRF-M	2.5	8	8	5	14	13	13	4.2	8.5
MRF-R	2.5	3	5	3	15	10	10	4.2	6.6
MRF-S-RV	13.5	13	11	15	2	8.5	8	13.8	10.6
MRF-M-RV	6.5	2	3	6	10	1	2	7.5	4.8
MRF-R-RV	6.5	1	2	9	9	6.5	9	7.5	6.3
MRF-N-S	10.5	15	14	12	3	14	7	13.8	11.2
MRF-N-M	2.5	5.5	7	4	13	12	14	4.2	7.8
MRF-N-R	2.5	7	6	2	12	2	4	4.2	5
MRF-N-J	10.5	14	15	13	5	11	11	13.8	11.7
MRF-N-S-RV	15.5	11	12	16	1	4	1	13.8	9.3
MRF-N-M-RV	6.5	4	4	7.5	7	8.5	6	4.2	6
MRF-N-R-RV	6.5	5.5	1	7.5	8	3	5	4.2	5.1
MRF-N-J-RV	13.5	10	10	14	4	6.5	3	13.8	9.4

Table 10: GG-FullC Rankings

GG-FullC	L	OA	UE	PB	Clr	VoD	HL	D	Avg
SPARF	1	12	1	3	1	3	16	2	4.9
SRF	6	1	8	16	8.5	15	15	5	9.3
MRF-S	9	4	16	12	11.5	13	11.5	10	10.9
MRF-M	15	10	4	1	5	14	6	9	8
MRF-R	13	16	7	6	14	12	4	15	10.9
MRF-S-RV	8	4	13	14.5	4	9.5	13	11	9.6
MRF-M-RV	10	6	5	7	3	7	7	7	6.5
MRF-R-RV	5	13	9	9	16	4	9	4	8.6
MRF-N-S	11	7	15	10	15	6	14	12	11.3
MRF-N-M	14	2	2	2	10	16	2	8	7
MRF-N-R	12	14	6	5	8.5	9.5	3	13	8.9
MRF-N-J	7	8	14	11	11.5	11	11.5	14	11
MRF-N-S-RV	3	9	11	13	13	8	10	1	8.5
MRF-N-M-RV	4	4	3	4	2	5	5	6	4.1
MRF-N-R-RV	2	15	8	8	5	2	8	3	6.4
MRF-N-J-RV	16	11	12	14.5	7	1	1	16	9.8

Averaging the ranks for all regression and classification tasks gives us the following scores for the different Random Forest predictors:

Table 11: GG-FullR + GG-FullC Rankings

GG-FullR+C	Avg
MRF-N-M-RV	5
MRF-M-RV	5.6
MRF-N-R-RV	5.7
MRF-N-R	6.9
MRF-N-M	7.4
MRF-R-RV	7.5
MRF-M	8.2
MRF-R	8.7
SPARF	8.7
MRF-N-S-RV	8.9
MRF-N-J-RV	9.6
MRF-S-RV	10.1
SRF	10.5
MRF-S	10.8
MRF-N-S	11.2
MRF-N-J	11.3

Based on these rankings we select the following multi-output Random Forests for further investigation: `MRF-M-RV`, `MRF-N-M-RV` and `MRF-N-R-RV`.

7.2 GG-Joint

The three selected multi-output Random Forests were run on three different constellations of classification and regression tasks, based on the GG-FullR and GG-FullC data sets. Tables 12-14 show the RMSE and accuracy scores of the three selected multi-output Random Forests and the two single-output Random Forests on the three joint GG datasets.

Table 12: GG-Joint Accuracy% and RMSE

GG-Joint	L	OA	UE	PB	Clr	VoD	HL	D
	Acc%	Acc%	Acc%	Acc%	RMSE	RMSE	RMSE	RMSE
SPARF	84.16	67.06	72.91	76.84	50.95	13.05	43.76	1.71
SRF	83.75	68.78	68.53	72.88	49.74	12.85	41.84	1.71
MRF-M-RV	84.84	66.13	71.53	75.59	49.14	12.12	41.62	1.69
MRF-N-M-RV	84.88	66.59	71.19	75.81	49.52	12.18	41.32	1.68
MRF-N-R-RV	84.59	66.94	71.44	75.50	49.13	12.33	41.71	1.69

Table 13: GG-Joint2 Accuracy% and RMSE

GG-Joint2	L	OA	UE	PB	Clr	VoD	HL	D
	RMSE	RMSE	RMSE	RMSE	RMSE	RMSE	RMSE	Acc%
SPARF	1.41	26.51	24.27	24.48	50.95	13.05	43.76	79.47
SRF	1.44	26.30	24.45	25.53	49.74	12.85	41.84	78.94
MRF-M-RV	1.42	26.41	24.38	25.80	50.03	12.30	42.10	79.25
MRF-N-M-RV	1.42	26.51	24.39	25.88	49.87	12.37	42.29	79.44
MRF-N-R-RV	1.43	26.47	24.42	25.90	49.51	12.29	41.52	79.16

Table 14: GG-Joint3 Accuracy% and RMSE

GG-Joint3	L	OA	UE	PB	Clr	VoD	HL	D
	Acc%	RMSE	RMSE	RMSE	RMSE	RMSE	RMSE	RMSE
SPARF	84.16	26.51	24.27	24.48	50.95	13.05	43.76	1.71
SRF	83.75	26.30	24.45	25.53	49.74	12.85	41.84	1.71
MRF-M-RV	84.13	26.41	24.35	25.93	49.73	12.17	41.15	1.69
MRF-N-M-RV	84.03	26.40	24.47	26.00	50.19	11.96	41.12	1.69
MRF-N-R-RV	84.19	26.32	24.34	25.77	49.91	12.27	41.61	1.69

We calculate the average ranks for these five Random Forests, and show here their average rank over: (1) all joint sets and (2) all joint sets, the full regression set, and the full classification set:

Table 15: Average GG Rankings

GG	J1+J2+J3	R+C+J{1,2,3}
MRF-M-RV	2.6	2.6
MRF-N-R-RV	2.7	2.7
MRF-N-M-RV	2.9	2.7
SPARF	3.3	3.2
SRF	3.5	3.8

7.3 BioPrint

The three selected multi-output Random Forests were also run on two version of the BP data set. In this trial, we compare the multi-output predictors only to (the higher ranking) `SPARF` Random Forest, and do not consider `SRF`.

Table 16: BP-FullR RMSE

BP-FullR				
Assay	SPARF	M-RV	N-M-RV	N-R-RV
100006	26.05	26.20	26.19	26.13
100029	20.07	20.25	20.22	20.16
100045	49.70	49.91	49.88	49.95
100077	11.15	11.26	11.26	11.27
100087	9.29	9.28	9.25	9.26
100115	17.42	17.73	17.75	17.68
100135	26.85	28.11	27.99	28.15
100160	16.32	16.42	16.43	16.40
100220	16.82	16.94	16.93	16.94
100221	19.71	19.97	19.95	19.96

Table 17: BP-Joint Accuracy% and RMSE

BP-Joint				
Assay	SPARF	M-RV	N-M-RV	N-R-RV
100006 Acc%	86.15	85.98	86.02	86.05
100029 Acc%	78.61	78.70	78.73	78.66
100045 Acc%	79.42	79.68	79.80	79.68
100077 Acc%	99.50	99.50	99.50	99.50
100087 RMSE	9.29	11.28	11.25	11.27
100115 Acc%	89.46	89.45	89.51	89.54
100135 Acc%	83.01	83.37	83.41	83.29
100160 Acc%	97.63	97.60	97.61	97.64
100220 Acc%	96.06	95.96	95.99	96.01
100221 Acc%	92.18	92.32	92.31	92.32

We calculate the task ranks and average ranks for each Random Forest for both the regression and joint data sets:

Table 18: BP-FullR Rankings

BP-FullR				
Assay	SPARF	M-RV	N-M-RV	N-R-RV
100006	1	4	3	2
100029	1	4	3	2
100045	1	3	2	4
100077	1	2.5	2.5	4
100087	4	3	1	2
100115	1	3	4	2
100135	1	3	2	4
100160	1	3	4	2
100220	1	3.5	2	3.5
100221	1	4	2	3
Avg	1.3	3.3	2.6	2.9

Table 19: BP-Joint Rankings

BP-Joint				
Assay	SPARF	M-RV	N-M-RV	N-R-RV
100006	1	4	3	2
100029	4	1	2	3
100045	4	2.5	1	2.5
100077	2.5	2.5	2.5	2.5
100087	1	4	2	3
100115	3	4	2	1
100135	4	2	1	3
100160	2	4	3	1
100220	1	4	3	2
100221	4	1.5	3	1.5
Avg	2.7	3.0	2.3	2.2

8 Analysis

This chapter covers analysis of the results presented in the previous chapter.

8.1 GG-FullR

For the eight regression tasks in the GG-FullR data set, we note a few interesting trends:

1. `SPARF` shows a better predictive performance for the Plasma Bound task than any of the tested multi-output variants, but for all other tasks, it fails to produce any high ranking results. `SRF` does not provide the best (or even high ranking) predictive performance for any of the tasks. We wish to remind the reader here that `SRF` uses the same underlying impurity measurement (information gain based on differential entropy) as `MRF` — in fact, the entropy calculations for `SRF` and `MRF` are identical, thus any difference in predictive performance between the two is a direct result of the information gain aggregation method.

2. In this full-regression set, normalization appears to have little effect on most tasks — for the Clearance task however, normalization provides an increased predictive performance ($+\,[0.03\%, 1.18\%]$) for all split functions; similarly, the Half-Life task sees either a small decrease ($-\,[0.41\%, 0.31\%]$) in predictive performance for the max gain function, but a relatively large increase ($+\,[1.34\%, 2.73\%]$) in performance for the sum and random gain functions.

   Table 20: Regression Performance From Normalization (%)

	L	OA	UE	PB	Clr	VoD	HL	D
MRF-S	0.06	-0.39	0.25	-0.11	1.08	-0.66	1.34	0.16
MRF-M	0.32	0.35	0.26	0.08	0.34	0.08	-0.31	-0.10
MRF-R	0.02	-0.34	-0.07	0.38	0.61	0.81	2.18	-0.27
MRF-S-RV	-0.19	0.25	-0.09	-0.15	0.12	0.35	2.73	0.19
MRF-M-RV	0.15	-0.08	-0.24	-0.07	1.18	-1.59	-0.41	0.39
MRF-R-RV	0.05	-0.31	0.28	0.37	0.03	0.38	2.08	0.25
Avg	0.07	-0.09	0.07	0.08	0.56	-0.11	1.27	0.10

   Nemenyi post-hoc test shows a statistically significant difference between `MRF-R-RV` and `MRF-N-R-RV` at $p = 0.1$. No statistical significance is detected for any other multi-output predictor.

3. Similar to normalization, random value selection affects tasks differently — however, for `-RV` the effects are much more pronounced. While the Oral Availability, Urinary Excretion, Clearance, Volume of Distribution and Half-Life tasks are mainly positively affected by random value splits, there is a clear negative effect on the LIPO, Plasma Bound and Distribution tasks.

Table 21: Regression Performance From Random Value Split (%)

	L	OA	UE	PB	Clr	VoD	HL	D
MRF-S	-1.75	-0.09	0.73	-0.53	1.24	-0.19	0.63	-0.37
MRF-M	-2.07	0.45	0.76	-0.15	1.69	1.95	3.80	-0.74
MRF-R	-2.43	0.23	0.52	-0.86	1.95	0.25	0.01	-0.76
MRF-N-S	-1.99	0.55	0.38	-0.56	0.28	0.83	2.02	-0.34
MRF-N-M	-2.23	0.02	0.26	-0.29	2.54	0.25	3.69	-0.26
MRF-N-R	-2.39	0.26	0.87	-0.87	1.37	-0.18	-0.09	-0.25
MRF-N-J	-1.76	0.33	0.74	-0.28	0.24	0.34	2.39	-0.26
Avg	-2.09	0.25	0.61	-0.51	1.33	0.46	1.78	-0.43

Nemenyi post-hoc test does not detect any significant difference in predictive performance due to random value selection for any multi-output Random Forest.

4. `MRF-M-RV` and `MRF-R-RV` (both normalized and non-normalized) always perform better than `SRF` on all tasks, and better than `SPARF` on all tasks except Plasma Bound — all other multi-output split functions perform worse than `SRF` on at least one task. All of the multi-output functions perform better than `SRF` on the LIPO, Volume of Distribution, Half-Life and Distribution tasks, and better than `SPARF` on the Oral Availability, Clearance, Volume of Distribution, Half-Life and Distribution tasks.

5. The max and random gain split functions show the most promising overall results, regardless of normalization and split value selection. The joint gain function does not provide the best result for any of the tasks. Interestingly, `MRF-N-S-RV` provides the best results for both Clearance and Half-Life despite sum gain showing a rather mediocre overall performance. Nemenyi post-hoc test shows a statistically significant difference in predictive performance between `MRF-M-RV`, `MRF-N-R` and `MRF-N-R-RV`; and `SPARF` and `SRF` at the following p-value cutoffs:

Table 22: P-value cutoffs GG-FullR

	SPARF	SRF
MRF-M-RV	0.1	0.5
MRF-N-R	0.1	0.1
MRF-N-R-RV	-	0.1

8.2 GG-FullC

We note the following for the classification trial:

1. `MRF` performs worse compared to `SPARF` on the classification version of this set — `SPARF` ranks highest for three tasks (LIPO, Urinary Excretion

and Clearance), and high for all other tasks except Half-Life. `MRF-N-R-RV` is the only multi-output variant that shows a better average rank than `SPARF`.

2. In this full-classification set, normalization again does not appear to have much effect; what small effect normalization does have is largely positive. `MRF-M` shows an interesting trade-off in accuracy between the Volume of Distribution and Half-Life tasks due to normalization.

Table 23: Classification Performance From Normalization (%)

	L	OA	UE	PB	Clr	VoD	HL	D
MRF-S	-0.14	-0.19	0.42	0.60	-0.27	1.02	-0.82	-0.36
MRF-M	0.27	0.74	1.55	-0.04	-0.94	-7.35	9.63	0.04
MRF-R	0.57	0.95	0.99	0.08	1.12	0.25	0.85	0.61
MRF-S-RV	0.65	-0.47	0.28	0.10	-1.56	0.04	1.02	1.92
MRF-M-RV	0.67	0.15	0.75	0.53	0.17	0.49	1.08	0.09
MRF-R-RV	0.19	-0.79	1.31	0.67	1.72	0.81	0.59	0.08
Avg	0.37	0.06	0.88	0.32	0.04	-0.79	2.06	0.40

Nemenyi post-hoc test shows a statistically significant performance increase from normalization for `MRF-R`, `MRF-R-RV` and `MRF-M-RV` at $p = 0.05$.

3. Random value selection too appears to generate an accuracy trade-off between tasks, albeit on a larger scale than normalization. `MRF-N-M` and `MRF-N-J` in particular show large accuracy trade-offs between tasks due to random value selection. The Clearance and Volume of Distribution tasks appear to be mainly positively affected by random value selection, but for the remaining tasks at least one of the split functions shows a distinct negative effect.

Table 24: Classification Performance From Random Value Split (%)

	L	OA	UE	PB	Clr	VoD	HL	D
MRF-S	0.04	0.00	1.20	-0.26	1.48	0.87	-0.32	-0.20
MRF-M	1.36	0.50	-0.17	-1.66	1.02	1.28	-0.40	0.15
MRF-R	1.48	1.42	-1.34	-2.10	-0.49	0.98	-1.86	2.43
MRF-N-S	0.83	-0.28	1.07	-0.76	0.17	-0.12	1.52	2.08
MRF-N-M	1.77	-0.09	-0.95	-1.11	2.15	9.85	-8.17	0.20
MRF-N-R	1.10	-0.33	-1.02	-1.52	0.10	1.55	-2.12	1.89
MRF-N-J	-6.19	-0.97	0.50	-0.60	0.98	8.91	13.48	-12.30
Avg	0.06	0.04	-0.10	-1.14	0.77	3.33	0.30	-0.82

Nemenyi post-hoc test does not detect any significant difference in predictive performance due to random value selection for any multi-output Random Forest.

4. Again, the max gain split function shows good results compared to other multi-output Random Forests, regardless of normalization. `MRF-R-RV` performs rather poorly, while `MRF-N-R-RV` shows competitive results — altough, `MRF-N-R-RV` is only able to improve the performance of `SRF` on 5 of the tasks. Nemenyi post-hoc test does not show any statistical significance when comparing the two single-output Random Forests to any of the multi-output variants.

8.3 GG-Joint

The joint classification-regression GG sets show similar trends as those we noted for the full regression and full classification sets:

1. The performance of `MRF` is generally better than both `SPARF` and `SRF` on regression tasks, but worse than `SPARF` on classification tasks. We also note that the relatively poor performance of `MRF` on the classification tasks appears to affect also the regression tasks negatively. The multi-output Random Forests — particularly `MRF-M-RV` — show better results than the single-output predictors on average, but for several decision tasks, all multi-output variants fail to produce better results than `SPARF`.
2. The multi-output Random Forests improve the predictive accuracy of `SRF` on all tasks except Oral Availability.
3. Predictive performance of the multi-output Random Forests is dependent on the composition of classification and regression tasks — i.e., the error of one task depends on the the type of the other tasks. This might be an indication that there is a significant difference in entropy calculations for classification and regression tasks despite normalization.
4. Nemenyi post-hoc test does not show any statistically significant difference in predictive performance between any pair of predictors at $p = 0.1$.

8.4 BioPrint

1. The two BP-sets show an effectively reversed effect for the multi-output Random Forests — here, the multi-output predictors compare favourably to `SPARF` on most classification tasks, but fail to beat `SPARF` on all but a single regression task.
2. Of the three multi-output Random Forests, the two normalized split functions show better results than the non-normalized split function on both the full regression set and the joint set.
3. Even though both `MRF-N-M-RV` and `MRF-N-R-RV` show better average rankings on the joint classification-regression set than `SPARF`, their prediction errors on the single regression task are increased compared to the full regression set. Nemenyi post-hoc test shows a statistical significance regarding the difference in predictive performance between `SPARF`; and `MRF-M-RV`

and `MRF-N-R-RV` at $p = 0.05$ on the regression data set. For the joint set, no statistical significance is detected.

9 Conclusions

The proposed method for combining classification and regression tasks within the same decision tree shows an improvement in predictive performance for most decision tasks compared to single-output decision trees based on the same node split functions (information gain using entropy or differential entropy). Compared to a Gini-index based decision trees however, performance of the proposed multi-output decision trees appears to be lacking.

The proposed method shows error rates that vary with the composition of classification and regression tasks in the data sets, even when entropies are normalized. Thus, we suspect that the proposed method considers classification and regression tasks differently — whether this is a result of properties inherent to the method or due to inaccurate approximation of differential entropies is undecided.

9.1 Contributions

We show that:

- The proposed method for producing joint classification-regression trees exhibits a better predictive performance than single-output decision trees based on the same node impurity metrics.
- Random information gain selection in the node-splitting of multi-output decision trees shows a similar predictive performance to max information gain, altough at a potentially lower computational cost.
- Random value selection (without consideration of output values) in the node splitting within multi-output decision trees can lead to an accuracy trade-off between tasks, where the error of one task is (sometimes greatly) reduced at the cost of an increased error of another task.
- Normalization of entropies in the node split function within multi-output decision trees provides a small but significant increase in predictive performance for most tasks.

9.2 Future Research

Based on our findings, we pose the folling questions as suggestions for future research:

- How does the proposed multi-output Random Forest perform on data sets where task relatedness is known *a priori*?
- Can the effects of random value selection be refined for multi-output problems (e.g. by considering output values), so that prediction error is reduced without any trade-off?

- Can random information gain selection be enhanced by applying it in a different manner (e.g. by randomly selecting a task for which to maximize information gain within each node)?
- Can the proposed method for calculating node impurities in joint classification-regression trees be improved by adding a correlation/covariance weight?

References

Boström, H. (2011). Concurrent learning of large-scale random forests. In *Proceedings of the Scandinavian Conference on Artificial Intelligence*, pages 20–29.

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

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and Regression Trees*. Chapman and Hall/CRC, 1 edition.

De'Ath, G. (2002). Multivariate regression trees: a new technique for modeling species-environment relationships. *Ecology*, 83(4):1105–1117.

Evgeniou, T. and Pontil, M. (2004). Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM.

Faddoul, J. B., Chidlovskii, B., Gilleron, R., and Torre, F. (2012). Learning multiple tasks with boosted decision trees. In *Machine Learning and Knowledge Discovery in Databases*, pages 681–696. Springer.

Faddoul, J.-B., Chidlovskii, B., Torre, F., and Gilleron, R. (2010). Boosting multi-task weak learners with applications to textual and social data. In *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*, pages 367–372. IEEE.

Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 148–156. MORGAN KAUFMANN PUBLISHERS, INC.

Gilman, A. G. et al. (1990). *Goodman and Gilmans' The pharmacological basis of therapeutics*. Springer.

Glocker, B., Pauly, O., Konukoglu, E., and Criminisi, A. (2012). Joint classification-regression forests for spatially structured multi-object segmentation. In *Computer Vision–ECCV 2012*, pages 870–881. Springer.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 28(1):75–105.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(8):832–844.

Krejsa, C., Horvath, D., Rogalski, S., Penzotti, J., Mao, B., Barbosa, F., Migeon, J., et al. (2003). Predicting adme properties and side effects: the bioprint approach. *Current opinion in drug discovery & development*, 6(4):470.

Larsen, D. R. and Speckman, P. L. (2004). Multivariate regression trees for analysis of abundance data. *Biometrics*, 60(2):543–549.

Linusson, H., Rudenwall, R., and Olausson, A. (2012). Random forest och glesa datarespresentationer.

Nemenyi, P. (1963). *Distribution-free multiple comparisons*. PhD thesis, Princeton University.

Quinlan, J. R. (1993). *C4. 5: programs for machine learning*, volume 1. Morgan kaufmann.

Segal, M. and Xiao, Y. (2011). Multivariate random forests. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):80–87.

Segal, M. R. (1992). Tree-structured methods for longitudinal data. *Journal of the American Statistical Association*, 87(418):407–418.

Tsoumakas, G. and Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13.

Zhang, H. (1998). Classification trees for multiple binary responses. *Journal of the American Statistical Association*, 93(441):180–193.

A Appendix

```
def train(dataset):
    root = Node(dataset)
    tree = grow_tree(root)

def grow_tree(parent):
    zs = parent.zs

    best_split = 0
    best_gain = 0
    best_split_val = 0

    for f_index in range(0, zs.m_features):
        for f_val in range(zs.min(feat), zs.max(feat)):
            left = zs.get(x: f_index(x) < f_val)
            right = zs.get(x: f_index(x) >= f_val)

            gain = calc_information_gain(zs, left, right)
            if gain > best_gain:
                best_gain = gain
                best_split = f_index
                best_split_val = f_val
                zs_left = left
                zs_right = right

    if (best_gain == 0):
        return # No split found, stop growing tree
    else:
        l = Node(zs_left)
        r = Node(zs_right)
        parent.add_left_child(grow_tree(zs_left))
        parent.add_right_child(grow_tree(zs_right))
```

Listing 1: C4.5.

```
def train(zs):
    trees = []

    for i in range(1, n_trees):
        bs = take_bootstrap(zs)
        root = Node(bs)
        tree = grow_tree(root)
        trees.append(tree)

def take_bootstrap(zs):
    sample = []
    while len(sample) < len(zs):
        sample.append(zs.take_random())
    return sample
```

Listing 2: Random Forest Induction.

```
def grow_tree(parent, m_try):
    features = []
    zs = parent.zs

    for i in range(1, m_try):
        f_index = random(1, zs.m_features)
        features.append(f_index)

    best_split = 0
    best_gini = Inf
    best_split_val = 0

    for f_index in features:
        split_gini, split_val, left, right = try_split(zs, f_index)
        if split_gini < best_gini:
            best_gini = split_gini
            best_split = f_index
            best_split_val = split_val

    if (best_gini == Inf):
        return # No split found, stop growing tree
    else:
        l = Node(zs.get(left.indices))
        r = Node(zs.get(right.indices))
        parent.add_left_child(grow_tree(l, m_try))
        parent.add_right_child(grow_tree(r, m_try))

def try_split(zs, f_index):
    best_split_val = 0
    best_split_gini = Inf

    for f_val in zs.inputs(f_index):
        left = zs.get(x: f_index(x) < f_val)
        right = zs.get(x: f_index(x) >= f_val)

        split_gini = calc_split_gini(zs, left, right)

        if split_gini < best_split_gini:
            best_split_gain = split_gini
            best_split_val = split_val
            zs_left = left
            zs_right = right

    return best_split_ini, best_split_val, zs_left, zs_right
```

Listing 3: Induction of CART with random subspacing.

Högskolan i Borås är en modern högskola mitt i city. Vi bedriver utbildningar inom ekonomi och informatik, biblioteks- och informationsvetenskap, mode och textil, beteendevetenskap och lärarutbildning, teknik samt vårdvetenskap.

På **institutionen Handels- och IT-högskolan (HIT)** har vi tagit fasta på studenternas framtida behov. Därför har vi skapat utbildningar där anställningsbarhet är ett nyckelord. Ämnesintegration, helhet och sammanhang är andra viktiga begrepp. På institutionen råder en närhet, såväl mellan studenter och lärare som mellan företag och utbildning.

Våra **ekonomiutbildningar** ger studenterna möjlighet att lära sig mer om olika företag och förvaltningar och hur styrning och organisering av dessa verksamheter sker. De får även lära sig om samhällsutveckling och om organisationers anpassning till omvärlden. De får möjlighet att förbättra sin förmåga att analysera, utveckla och styra verksamheter, oavsett om de vill ägna sig åt revision, administration eller marknadsföring. Bland våra **IT-utbildningar** finns alltid något för dem som vill designa framtidens IT-baserade kommunikationslösningar, som vill analysera behov av och krav på organisationers information för att designa deras innehållsstrukturer, bedriva integrerad IT- och affärsutveckling, utveckla sin förmåga att analysera och designa verksamheter eller inrikta sig mot programmering och utveckling för god IT-användning i företag och organisationer.

Forskningsverksamheten vid institutionen är såväl professions- som design- och utvecklingsinriktad. Den övergripande forskningsprofilen för institutionen är handels- och tjänsteutveckling i vilken kunskaper och kompetenser inom såväl informatik som företagsekonomi utgör viktiga grundstenar. Forskningen är välrenommerad och fokuserar på inriktningarna affärsdesign och Co-design. Forskningen är också professionsorienterad, vilket bland annat tar sig uttryck i att forskningen i många fall bedrivs på aktionsforskningsbaserade grunder med företag och offentliga organisationer på lokal, nationell och internationell arena. Forskningens design och professionsinriktning manifesteras också i InnovationLab, som är institutionens och Högskolans enhet för forskningsstödjande systemutveckling.

HÖGSKOLAN I BORÅS
VETENSKAP FÖR PROFESSION

BESÖKSADRESS: JÄRNVÄGSGATAN 5 · POSTADRESS: ALLÉGATAN 1, 501 90 BORÅS
TFN: 033-435 40 00 · E-POST: INST.HIT@HB.SE · WEBB: WWW.HB.SE/HIT