Tree regression models using statistical testing and mixed integer programming

Abstract: Regression analysis is a statistical procedure that fits a mathematical function to a set of data in order to capture the relationship between dependent and independent variables. In tree regression, tree structures are constructed by repeated splits of the input space into two subsets, creating if-then-else rules. Such models are popular in the literature due to their ability to be computed quickly and their simple interpretations. This work introduces a tree regression algorithm that exploits an optimisation model of an existing literature method called *Mathematical Programming Tree* (`MPtree`) to optimally split nodes into subsets and applies a statistical test to assess the quality of the partitioning. Additionally, an approach of splitting nodes using multivariate decision rules is explored in this work and compared in terms of performance and computational efficiency. Finally, a novel mathematical model is introduced that performs subset selection on each node in order to select an optimal set of variables to considered for splitting, that improves the computational performance of the proposed algorithm.

Keywords: Mathematical programming, Regression analysis, Decision trees, Subset selection, Optimisation

1 Introduction

Regression analysis is a predictive modelling technique for formulating the correlation between a set of dependent and independent variables. Depending on how the values of the independent variables (i.e. predictors) vary, the value of the dependent variable (i.e. response) also changes. The objective of regression is to apply a mathematical function to the data that captures these changes.

There are numerous algorithms in the literature for regression, with varying degrees of complexity. Examples include linear regression, Support Vector Machine Regression (`SVM`) (Cortes and Vapnik, 1995), K-nearest neighbors (`KNN`) (Cover and Hart, 1967) and Multivariate Adaptive Regression Splines (`MARS`) (Friedman, 1991).

Ensemble methods have gained a lot of interest due to their ability to produce good results and can be divided into two major groups, sequential and parallel methods. Parallel methods create multiple 'weak learners' in parallel and combine them to achieve better predictive performance than creating a single 'strong learner'. One such popular algorithm is Random Forest (Breiman, 2001). Sequential methods are similar, meaning that multiple models are created. However, those models are trained sequentially and each model focuses on where the previous one performed poorly. After many iterations, the result is a single model with much better predictive performance than the original. Popular methods include Adaptive Boosting (`AdaBosot`) (Freund and Schapire, 1997) and `XGBoost` (Chen and Guestrin, 2016).

In the field of mathematical optimisation there is the Automated Learning of Algebraic Models for Optimization (`ALAMO`) (Cozad et al., 2014; Wilson and Sahinidis, 2017), a mixed integer optimisation approach called Classification and Regression via Integer Optimisation (`CRIO`) (Bertsimas and Shioda, 2007), as well as two piecewise regression approaches called Optimal Piecewise Linear Regression Analysis (`OPLRA`) (Yang et al., 2016) and Piecewise Regression with Optimised Akaike Information Criterion (`PROA`) (Gkioulekas and Papageorgiou, 2018, 2019).

This work focuses on generating tree regression models. Those models are conceptually simple yet powerful. Tree structures are constructed by repeated splits of the input variables into two descendant subsets, called child nodes. These splits create `if-then-else` rules to assign samples into child nodes. Those nodes that are not split any further are called terminal or leaf nodes and each leaf has an explicit regression model. Popular tree algorithms include *Classification and Regression Trees* (`CART`) (Breiman et al., 1984), `C4.5` (Quinlan, 1993) and others.

Advances have been made in the field of integer optimisation as well, with tree approaches being able to handle both classification and regression tasks. An algorithm called (`DTIP`) (Verwer and Zhang, 2017) has been developed that uses integer programming to construct trees of certain depth from data sets of sizes up to 1000 samples.

Multivariate decision trees have also been an intriguing research field. Multivariate decision trees mainly differ from univariate tree by creating decision rules on more than one attributes (typically a linear combination of attributes) (Kotsiantis, 2013). Examples include Brodley and Utgoff (1995), Sok et al. (2015), Wang et al. (2018) and Wang et al. (2020). Furthermore, Bertsimas and Dunn (2017) developed an algorithm called Optimal Classification Trees (`OCT`) that uses mixed integer optimisation techniques in order to generate tree structures.

In previous work, Yang et al. (2017) developed a tree regression algorithm that employed a mathematical programming model to optimise the value of the break point when splitting nodes. That optimisation model is solved iteratively for every input variable in order to identify the optimal split. The algorithm also used a heuristic approach to control the tree generation process and ulti-

mately the size of the tree. That heuristic was based on the introduction of a parameter that was set to a fixed value based on a sensitivity analysis. As a result, the algorithm has no parameters that require tuning.

However, when new examples are introduced to this algorithm, the predictive results are often poor. This work addresses the need for a new stopping criterion to control the size of the generated tree. By accommodating a well established statistical test, the new proposed algorithm is able to balance prediction accuracy and model complexity.

Furthermore, the path of creating multivariate decision rules is also explored. The optimisation model for splitting nodes is modified to address the new rules. In addition to that, a novel mathematical formulation is proposed to create a subset selection model to enable the algorithm to search the optimal partitioning variable based on a reduced set of input variables.

2 Proposed regression tree algorithm

This section proposes three variations of a novel regression tree algorithm that is based on the partitioning model of appendix A. That model is part of the `MPtree` algorithm that is mentioned in Section 1 and it is responsible for optimally splitting a node into two nodes while minimising the absolute deviation of the fitting.

The first variant of the algorithm uses the same optimisation model, but the criterion to assess node splittings and ultimately terminate the tree generation process is substituted with the Chow statistical test. The second variant uses a slightly modified version of the optimisation model to create decision rules based on linear expressions instead of a fixed value based on a particular input feature. The last variant introduces a new mathematical formulation that minimises the value of the *Bayesian Information Criterion* (*BIC*) to select an optimal subset of input variables to be considered for splitting.

2.1 Variant I: Introduction of a statistical test

2.1.1 The Chow statistical test

In regression, the F-test can be used to assess the quality of stepwise regression. Assuming there is a break in the data, splitting them into two subsets and fitting separate regression models to each one can result in a better overall fit . However, this action adds to the complexity of the model and might lead to overfitting. Therefore, the Chow test can be applied to compare the predictive performance of a segmented and a non-segmented regression model.

Suppose that there are two subsets and the question is whether to perform regression on the entire dataset consisting of both subsets (we denote this model 1), or to apply separate regression models for each subset (we denote this model 2). So RSS_1 is the residual sum of squares for the first model and RSS_2 is the residual sum of squares for model 2 (which in this case is the sum of the RSS for each subset). In general, there will be an improvement when splitting the data ($RSS_2 \leq RSS_1$), with equality occurring only when all the regression coefficients for the two models coincide (Dougherty, 2011). However, there is a trade-off due to the added complexity of the overall regression model. By splitting the data into two subsets and performing separate regressions, more parameters are added to the model and hence more degrees of freedom. So the chow test is useful for testing if there is a statistically significant difference in predictive performance.

The F statistic for the chow test can be computed as follows (Dougherty, 2011):

$$F = \frac{\left(\frac{RSS_1 - RSS_2}{p_2 - p1}\right)}{\frac{RSS_2}{n - p_2}} \tag{1}$$

where:

RSS_1	residual sum of squares of model 1 (single regression for the entire dataset)
RSS_2	total residual sum of squares of model 2 (separate regression for each subset)
p_1	regression parameters of model 1
p_2	regression parameters of model 2
n	number of samples in the dataset

The null hypothesis states that model 2 **does not** provide a significantly better fit than model 1. So the procedure to either reject or accept the null hypothesis is as follows:

- Calculate the F statistic using equation 1
- Choose an appropriate confidence level (e.g. 99%)
- Calculate the critical F_{crit} value of the F-distribution
- Reject the null hypothesis if $F > F_{crit}$

According to the steps above, if there is evidence to reject the null hypothesis, it is accepted that model 2 **does** provide a significant improvement in predictive performance. The larger the F statistic is, the better the overall fit.

2.1.2 Application to tree regression

The use of this test can aid the process of generating regression trees since it can be used as a criterion for splitting nodes. Every node can be considered a population that can be split into two separate subsets and a decision has to be made of either accepting or rejecting the splitting.

The main steps of the algorithm for constructing trees are given below. As with other tree algorithms, recursive splitting is used to generate the tree. For each node, the partitioning model of appendix A is applied to split it into two child nodes. Then the Chow test is applied, comparing the linear regression model with the segmented regression model. If there is a significantly better predictive performance by splitting the node, then the partitioning is approved and the algorithm starts again by following the same procedure for all the new nodes. However, if a node splitting is rejected then this node will no longer be considered for splitting. The entire tree generation process is terminated when there are no more nodes that are eligible for splitting.

In previous work, the proposed `MPtree` algorithm used a heuristic approach to control the tree generation process. This heuristic introduced a new parameter which was used as a threshold to the reduction percentage of the absolute deviation. That reduction in error was a comparison between the current examined node and the root node. By performing a sensitivity analysis, the authors concluded that the parameter should be set to the value of 0.015, as it yielded the best results.

The proposed algorithm, from now on called `StatTree`, is briefly explained below.

- Exhaustive search over the entire set of input variables. Apply the partitioning model of appendix A to split every variable into two child nodes.
- Identification of the optimal partitioning variable. This variable is the one that has the minimum fitting error from the previous step.
- Assess the quality of the splitting by performing the Chow test. If the hypothesis is rejected, then the splitting is approved and the new nodes can be considered candidates for further splitting. Otherwise, the current node becomes a leaf node.
- Repeat those step until there are no more candidate nodes left to be checked.

2.2 Variant II: Mutlivariate decision splittings

As mentioned in section 1, multivariate decision trees generate rules as that are a combination of multiple features. Such a complex rule could be a better representation of the data and improve performance. There is a trade-off however between performance and model interpretability.

A decision tree with multivariate splitting rules can be obtained by modifying the optimisation model of appendix A. The new additions to the notation for this work are presented below:

Continuous variables

$\hat{a}_m$	coefficients for splitting instances
$\hat{b}$	parameter for splitting instances

The new added variables are $\hat{a}_m$ and $\hat{b}$, which represent the coefficients and the breaking value of the rules respectively.

Mathematical Constraints

Constraints 18 and 19 in the optimisation model of appendix A are the only ones that require modification to address the new decision rule changes. The modifications are presented below:

$$\sum_m a_{sm} \cdot \hat{a}_m \leq \hat{b} + u_1 \cdot (1 - F_{sc}) - \epsilon \qquad \forall\ s \in S_n,\ c = left \tag{2}$$

$$\hat{b} - u_1 \cdot (1 - F_{sc}) + \epsilon \leq \sum_m a_{sm} \cdot \hat{a}_m \qquad \forall\ s \in S_n,\ c = right \tag{3}$$

The rest of the constraints remain the same. This optimisation model is integrated to the rest of the proposed `StatTree` algorithm. This variant is called $\texttt{StatTree}_{\texttt{LF}}$ (`StatTree` with Linear Functions).

2.3 Variant III: Variable selection for splitting nodes

According to section (2.1.2), identifying the best partitioning variable employs an exhaustive search approach. Multiple *Mixed Integer Linear Programming* (*MILP*) models are solved iteratively in order to select the partitioning variable that yields the minimum error. This approach adds to the overall computational time of training a model, hence making the use of this algorithm on datasets with a large number of variables impractical.

Feature or subset selection methods are useful for determining a smaller subset of the original input space, leading to potential improvement in prediction accuracy and interpretability of the final model. In this section, a feature selection method will be employed in order to identify a subset

of the original input variables to consider for splitting. So the task at hand is to formulate a model that decides on the optimal subset of variables. One way of dealing with such problems is the use of information criteria such as the *Akaike Information Criterion* (*AIC*) and the *Bayesian Information Criterion* (*BIC*). These two criteria have been established as two of the most frequently used in the literature with a wide variety of applications.

Jian et al. (2017), proposed a variable selection method through mixed integer quadratic programming by using utilising BIC, whereas Miyashiro and Takano (2015) developed a mixed integer programming approach to deal with the problem of subset selection using Mallows' C_p. For this work, a feature selection model will be developed using the BIC as an objective function. However, for simplicity reasons this approach will be formulated as an *MILP* approach.

The general formulation of the *BIC* is as follows (Wagenmakers and Farrell, 2004):

$$BIC = -2 \cdot \ln(\hat{\mathcal{L}}) + K \cdot \ln(n) \tag{4}$$

where:

ln	natural logarithm
$\hat{\mathcal{L}}$	Value of the log-likelihood function at its maximum point
K	number of parameters in the model
n	number of samples in the data

In regression analysis, if all the candidate models assume normally distributed errors with a constant variance, then the criteria can be reformulated as (Burnham and Anderson, 2003):

$$BIC = n \cdot \ln\left(\frac{RSS}{n}\right) + K \cdot \ln(n) \tag{5}$$

where:

RSS	residual sum of squares

Given a set of candidate models the criterion can be used for model selection. The model that achieves the minimum *BIC* value gets picked as the best performer. It is clear that the use of the *BIC* poses a problem since the formulation is non-linear. Formulating the problem as an *MILP* requires adjustments. This formulation is based on a recent work by Gkioulekas and Papageorgiou (2019), in which there are two simplifications. First, the logarithm function is approximated by using piecewise linear expressions and second, the use of absolute deviation is used instead of *RSS*. These two steps ensure a linear formulation of the criterion. The mathematical formulation is described below:

Indices

s	data samples, s=1,2,...,S
m	feature/independent input variable
i	number of breaking points, i=1,2,...,N

Parameters

a_{sm}	numeric value of sample s on feature m
y_s	output value of sample s
LO	lower limit for the linear regression coefficients
UP	upper limit for the linear regression coefficients
γ_i	discrete points for the linearisation
β_i	output values of the discrete points
N	maximum number of variables to be selected by the model

Continuous variables

B	intercept of regression function in child node c
D_s	training error between predicted output and real output for sample s
Pr_s	predicted output for sample s in child node c
W_m	regression coefficient for feature m in child node c
BIC	Bayesian Information Criterion value
G	result of the approximation

Binary variables

Z_m	1 if feature m is selected; 0 otherwise

SOS2 variables

λ_i	Variables that describe which discrete points will be used for the linear approximation

Mathematical Constraints

This mathematical formulation fits a linear regression model to a set of data according to the following constraint

$$P_s = \sum_m a_{sm} \cdot W_{mc} + B \qquad \forall\ s \tag{6}$$

The absolute deviation between the observed values and model predictions is formulated by the following pair of equations.

$$D_s \geq y_s - P_s \qquad \forall\ s \tag{7}$$

$$D_s \geq P_s - y_s \qquad \forall\ s \tag{8}$$

This formulation introduces a set of binary variables (Z_m) to achieve feature selection. For every feature that is selected, the corresponding variable takes the value of $Z_m = 1$, otherwise $Z_m = 0$. By restricting the values of the coefficients between specified upper and lower bounds, if a variable m is not selected, then the regression coefficient will be forced to zero. Otherwise, the coefficient can take any value between the bounds. By setting very large positive and negative values to those bounds, the regression coefficients are essentially free to take any real value if the corresponding features are selected.

$$W_m \geq LO \cdot Z_m \qquad \forall\ m \tag{9}$$

$$W_m \leq UP \cdot Z_m \qquad \forall\ m \tag{10}$$

Logical constraints are formulated to ensure that at least one variable has to be selected in the final regression model and that a maximum number of N variables can be selected. N is a user specified parameter that can take integer values. This constraint enables the user to control the size of the selected subset.

$$\sum_m Z_m \geq 1 \tag{11}$$

$$\sum_m Z_m \leq N \tag{12}$$

Due to the non-linear nature of the *BIC*, it is desirable to reformulate the criterion by using piecewise approximations and construct the model as *MILP* that can be solved to optimality. Equations 13-16 approximate the logarithm function through piecewise linear expressions. Variable λ_i is a SOS2 variable (special ordered set of type 2), which means that at most two λ variables can take non-zero values and those values have to be for adjacent variables in that set.

$$\beta_i = \ln \gamma_i \qquad \forall\ i \tag{13}$$

$$\sum_s D_s = \sum_i \gamma_i \cdot \lambda_i \tag{14}$$

$$G = \sum_i \beta_i \cdot \lambda i \tag{15}$$

$$\sum_i \lambda_i = 1 \tag{16}$$

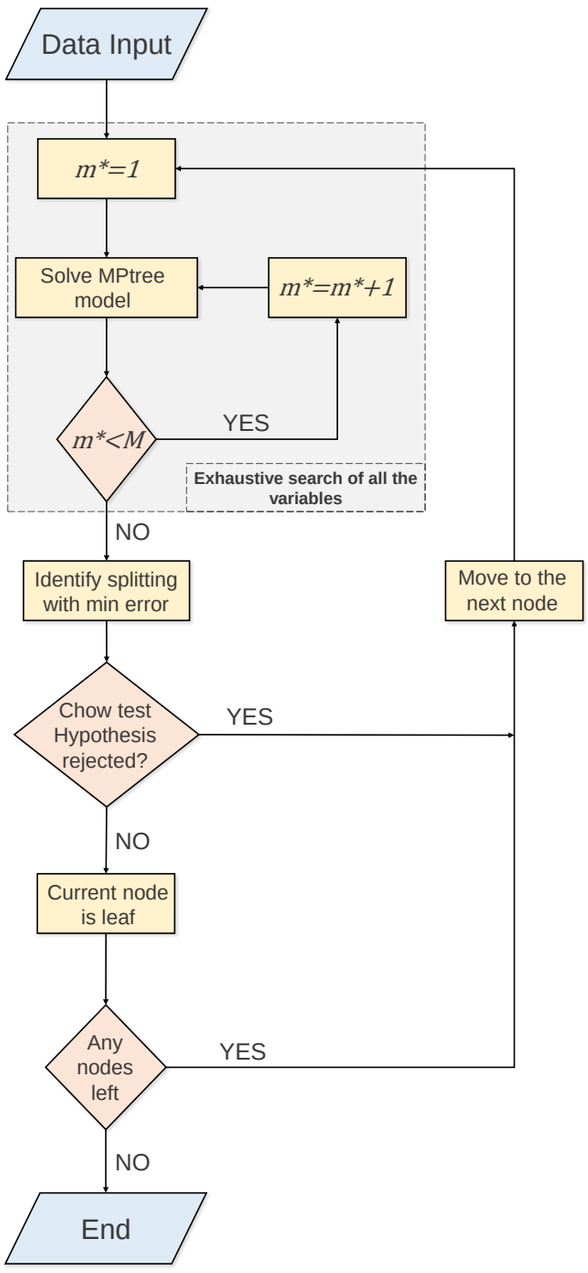
The objective function is the value of *BIC* which is formulated as follows:

$$\min\ BIC = |S| \cdot G - |S| \cdot \ln |S| + \ln |S| \cdot \left(\sum_m Z_m + 1 \right) \tag{17}$$

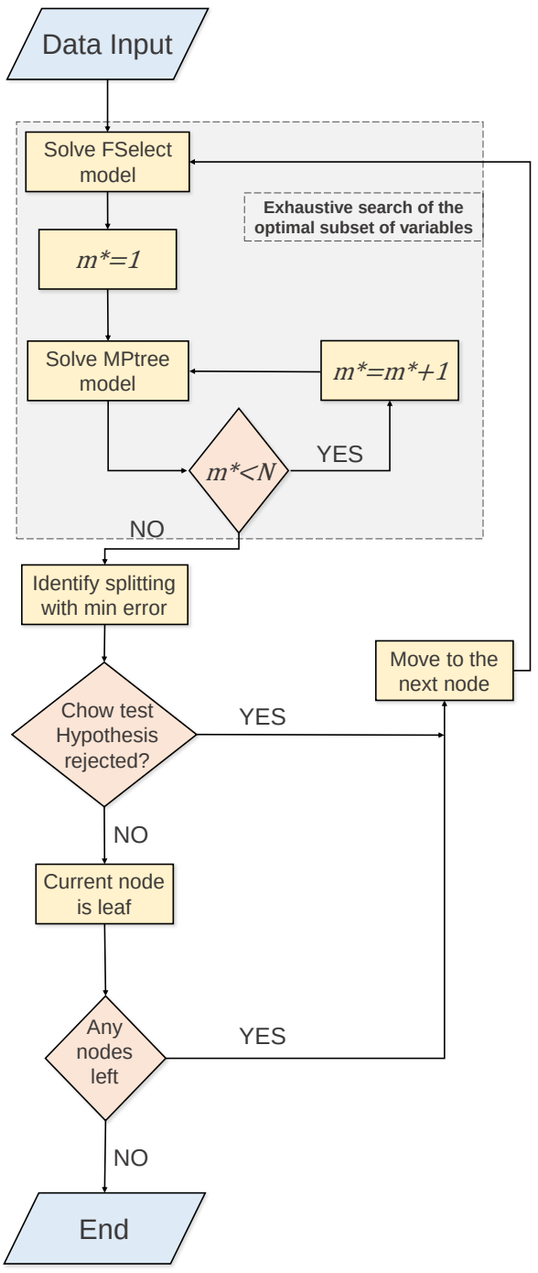
$|S|$ is the total number of samples in the dataset, G is the piecewise linear approximation of the logarithm. The first term of the right-hand side of Equation 17 is the fitting error, whereas the last term the last term is the complexity of the model. The complexity of the model is the total number of selected variables ($\sum_m Z_m$) and an extra degree for the intercept of the regression.

The resulting model, from now on known as `FSelect`, can be summarised as:
minimise (17)
subject to (6)-(16) constraints
and is formulated as an *MILP* problem that can be solved to optimality.

A maximum number of selected variables is specified by the user with the final number being decided by the optimisation model since the *BIC* tries to find a balance between prediction accuracy and model complexity.



(a) `StatTree` - Exhaustive search



(b) `StatTree` with `FSelect`

Figure 1: The proposed variants. Case (a) is the exhaustive search based on all the variables. Case (b) selects a subset of size N to consider for splitting.

Figure 1 is a description of two variants. The fundamental difference between the two variants is the inclusion of the `FSelect` model that eliminates the exhaustive search of the entire input space. However, the step to identify a partitioning variable is still required. The $\texttt{StatTree}_{\texttt{LF}}$ variant completely eliminates this step, since the decision rule is multivariate linear expression.

3 Computational part

3.1 Examined datasets

A number of examples are considered in this work which are summarised in Table 1. Those datasets are derived from different online sources. Specifically the pharmacokinetics, earthquake, abalone and speeding datasets are available through a package in `R`, boston, bodyfat and sensory

datasets are available through *StatLib* (Vlachos, 2005), concrete, cooling, heating, wine and yacht through the *UCI* machine learning repository (Dheeru and Karra Taniskidou, 2017) and the rest are available through the *KEEL* dataset repository (Alcalá-Fdez et al., 2011).

Table 1: Regression datasets examined in this work

Data	**Predictors**	**Samples**	**Data**	**Predictors**	**Samples**
Concrete	8	1030	Octane	4	82
Cooling	8	768	Pharma	4	132
Heating	8	768	Plastic	2	1650
Yacht	6	308	Sensory	11	576
Bodyfat	14	252	Wankara	9	1609
Boston	13	506	Abalone	8	4177
Dee	6	365	Speeding	3	8437
Earthquake	4	1000			

The `Yacht` hydrodynamics set predicts the residuary resistance of sailing yachts for evaluating the ships' performance and for estimating the required propulsive power. An assessment of heating and cooling load requirements is captured in the `Energy Efficiency` dataset (Tsanas and Xifara, 2012), of different buildings as a function of 8 parameters. The `Concrete` dataset (Yeh, 1998) predicts the compressive strength of concrete as a structural material.

A study of the kinetics of the anti-asthmatic drug theophylline is included in the `Pharma` dataset. Twelve subjects were given oral doses of the drug and the aim is to predict the final theophylline concentration of each subject by measuring parameters such as weight and time. Earthquake data based on the location of seismic events that occurred near Fiji since 1964 are in the `earthquake` dataset. The `Bodyfat` dataset uses features such as age, weight and height to measure the percentage of bodyfat in a subject. An evaluation of wine quality by 6 judges is recorded in the `Sensory` dataset.

`Dee`, predicts the daily average price of electricity in Spain. The dataset contains values about the daily consumption of energy from various sources such as hydroelectric, fuel, natural gas and more. `Plastic`, computes how much pressure can a given piece of plastic withstand when a force is applied on it at a fixed temperature. `Wankara`, contains observations about weather information of Ankara during 1994-1998, with the goal of predicting the average temperature. `Abalone`, predicts the age of abalone from physical measurements which are easy obtain. The `Speeding` dataset has been collected from a study that tried to identify the effect of warnings signs on speeding patterns. The speed measurements were taken before the erection of a warning sign, after shortly after the erection of the sign and finally after the sign had been in place for some time. Finally, `Boston` consists of observations that predict the price of houses in various places in Boston.

3.2 Validation of the method

The simplest way to evaluate a model is to split the original data into two subsets, one for training and one for testing. The training set will be used to construct a regression model, which will be evaluated by using the testing set. The reason for doing so is to measure how well the model generalises to new, previously unseen data.

Cross-validation (CV) is a statistical method of evaluating the performance of models. The most common form of cross-validation is k-fold where the data is split into k subsets of equal size. Then the method uses one of these sets for testing and the rest for training. The method stops when all

of the k sets have been used as the testing set. Parameter k is user-specified and is usually set to either 5 or 10 (Muller and Guido, 2016).

In this work, 5-fold cross-validation is selected to evaluate the performance of the proposed algorithm. 10 runs are performed and the *Mean Absolute Error* (*MAE*) between model prediction and the observed values is calculated for each fold. The final score is the average of all the runs. The algorithm is implemented in the `R v.3.3.1` programming language (R Development Core Team, 2016), while the mathematical programming model responsible for partitioning the nodes, is implemented in the *General Algebraic Modeling System* (`GAMS`) `v.24.7.1` (GAMS Development Corporation, 2016) and solved using the `CPLEX v.12.6.3.0` solver with optimality gap set at 0 and a time limit of 200s. `R` is also used for the k-fold cross-validation procedure by utilising the `caret` package (Kuhn, 2008).

A number of tree methods from literature are also implemented in this work for comparison purposes on the same datasets. All of those methods are implemented in `R` using the appropriate packages. The methods include *Classification and Regression Trees*, `CART`, (Breiman et al., 1984) using the `rpart` package (Therneau et al., 2018), `M5P` regression (Quinlan et al., 1992; Wang and Witten, 1996) using the `RWeka` package (Hornik et al., 2009; Witten et al., 2016), *Conditional inference trees*, `CTree` , (Hothorn et al., 2006), using the `partykit` package (Hothorn and Zeileis, 2015), `Cubist` using the `Cubist` package (Kuhn and Quinlan, 2017) and `MPtree` (Yang et al., 2017) which was implemented in `R` and `GAMS`. The same 10 runs of 5-fold cross-validation are performed to evaluate and compare with the proposed methods.

4 Results

4.1 Variant I: Introduction of a statistical test

It is common practice to pre-process and normalise the data in order to avoid having input variables being more dominant than others in the final regression. For this work, the `BBmisc` package is used in `R` (Bischl et al., 2017) to perform feature scaling and normalise the input variables to the range of [0,1].

4.1.1 Cross-validation results

Table 2 contains the *MAE* results of all the runs of cross-validation. For each dataset, the method that performed the best is marked with bold. `StatTree` has the best performance in terms of *MAE* score for 7 out of 15 examples. `Cubist` is the next best performer with 3 out 15, `MPtree` and `M5P` 2 out 15 and `CART` with only a single dataset. However, that alone is not a good indication of overall performance.

Table 2: Cross-validation results of `StatTree` using *MAE*

	StatTree	MPtree	Cubist	CART	M5P	CTree
Concrete	4.329	4.868	**4.267**	7.239	4.656	5.295
Cooling	1.175	**0.891**	0.938	2.400	1.210	1.403
Heating	0.367	0.354	**0.347**	2.011	0.693	0.665
Yacht	**0.539**	**0.539**	0.557	1.669	0.931	0.802
Bodyfat	**0.183**	5.282	0.205	1.356	0.373	0.911
Boston	2.568	4.644	2.587	3.234	**2.501**	3.014
Dee	**0.313**	0.975	0.316	0.381	0.316	0.356
Earthquake	7.345	12.427	7.294	8.223	**7.273**	7.884
Octane	0.391	0.805	**0.384**	0.602	0.464	0.591
Pharma	0.900	**0.870**	1.053	1.339	1.328	1.566
Plastic	**1.226**	1.230	1.229	1.658	1.234	1.410
Sensory	0.610	0.663	0.602	**0.578**	0.601	0.593
Wankara	**0.972**	3.605	1.000	3.213	0.977	1.574
Abalone	**1.490**	1.512	1.500	1.731	1.521	1.600
Speeding	**4.143**	4.243	4.188	4.524	4.239	4.581

Constructing a figure to visualise the comparison of the various methods will aid the interpretation of the overall predictive performance. In this figure, for each dataset the best performer is awarded 10 points whereas the worst performer is awarded 1 point. The final ranking is the average score that each method achieves across all datasets.

Looking at Figure 2 makes it is easier to compare the overall performance of the methods. The `StatTree` algorithm is ranked at number 1. Also, a large performance gap exists between `StatTree` and `MPtree`, which indicates that the new proposed method is actually a better alternative. `Cubist` on the other hand, is the only method that can provide competitive results. However, since the performance of those two methods is very close, a statistical test has to be applied in order to check whether there is a significant difference in the results.

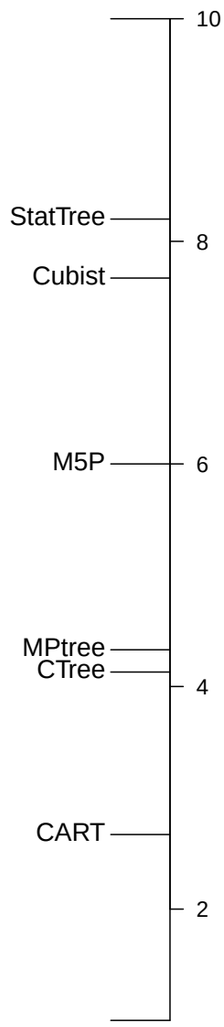
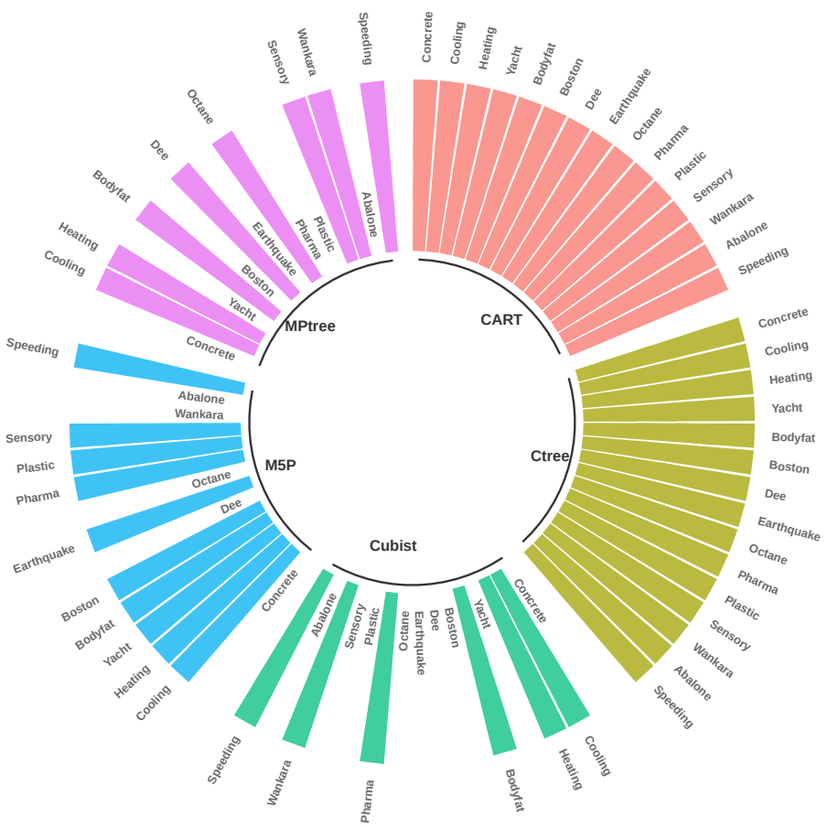


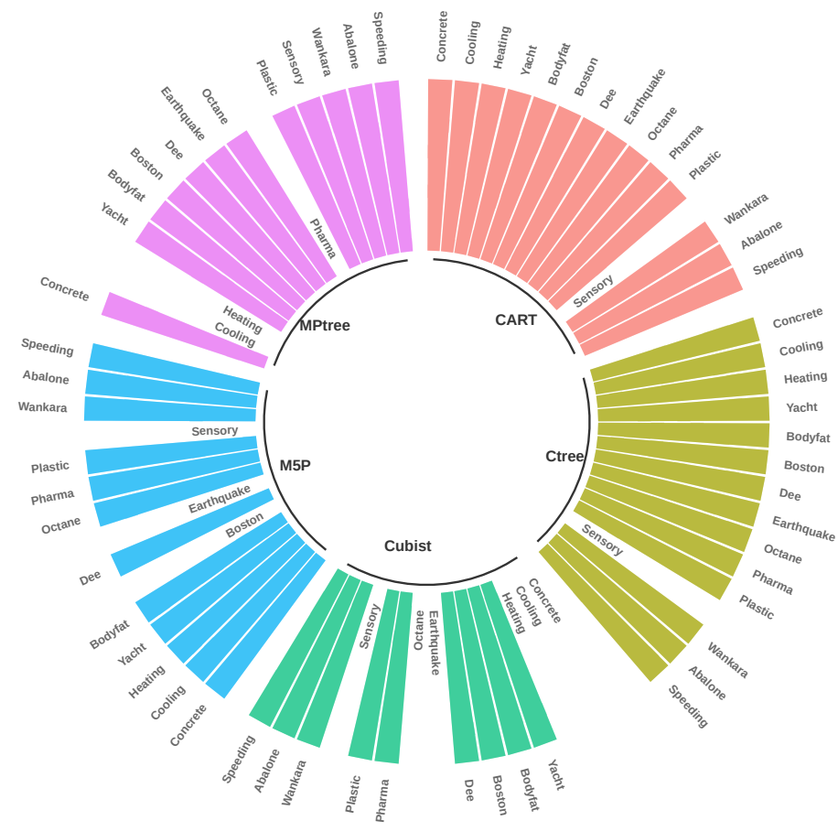
Figure 2: Visualisation of the performance of the methods based on the MAE results

4.1.2 Statistical analysis

For this statistical analysis the Welch's t-test is selected. This is a two-sample test which is used to test the hypothesis that two populations have equal means and is reliable when the samples have unequal variances (Welch, 1947). If we have evidence to reject this hypothesis, then we can conclude that the difference between the two means is significant.



(a) Welch's t-test



(b) MAE scores

Figure 3: Visualisation of the computational results of the `StatTree` variant. Case (a) is a representation of the statistical analysis using Welch's t-test. Case (b) is a representation of the MAE performance.

For each dataset, the two different populations that will be compared are the values of the 10 cross validation runs between the `StatTree` algorithm and one of the rest. If by performing the Welch's t-test there is evidence to reject the null hypothesis, then it can be concluded that there is a statistically significant difference in the two sample means and the best method is the one that

has the minimum average error.

More information about the calculation of the t statistic and the degrees of freedom is available at appendix B. The application of the t-test follows a similar procedure as the one described for the F-test in section (2.1.1):

- Calculate the t statistic using equation 25
- Choose a confidence level of 99% ($\alpha = 0.01$)
- Calculate the probability p-values of the t distribution
- Reject the null hypothesis if $p < a$

Figure 3 is a visual representation of the statistical analysis performed for the CV results. The circles contain five groups, one for each competing tree regression algorithm. Each group has 15 bars that correspond to the 15 examined datasets. Figure 3a is a visualisation of the statistical analysis that has been performed to compare the regression methods. A bar is present only if there is a significant statistical difference in the results (null hypothesis of the t-test is rejected). For all the examined examples, there is a difference with `CART` and `CTree`. There is a significant difference in 11 and 8 examples with `M5P` and `MPtree` respectively. With `Cubist` however, there is a difference in only 6 examples.

Figure 3b is similar to 3a, but this time a bar is present only if `StatTree` has achieved a lower MAE score for this specific example. It is clear that the proposed approach has consistently outperformed `CART`, `CTree` and `M5P`. Even though `MPtree` is based on the same optimisation model for splitting nodes, the addition of the F-test has greatly improved the results, with `StatTree` providing better MAE score in 12 out of the 15 examples.

To accurately compare those algorithms, both figures should be taken into account. So, for a specific example it is desirable to have a bar present in both figures. If that is the case, there is strong evidence to suggest that the MAE averages of all the CV runs are indeed different and `StatTree` provides better error values than the competitor. Based on that rule, the proposed algorithm provides better predictive results than `CART`, `CTree` and `M5P`. It also provides better predictive results than `MPtree` in 6 out of 8 examples (there is a significant statistical difference in 8 examples). `StatTree` is very competitive against `Cubist` as well and despite having a statistical difference in only 6 examples, it has lower error values in 4 out of those 6.

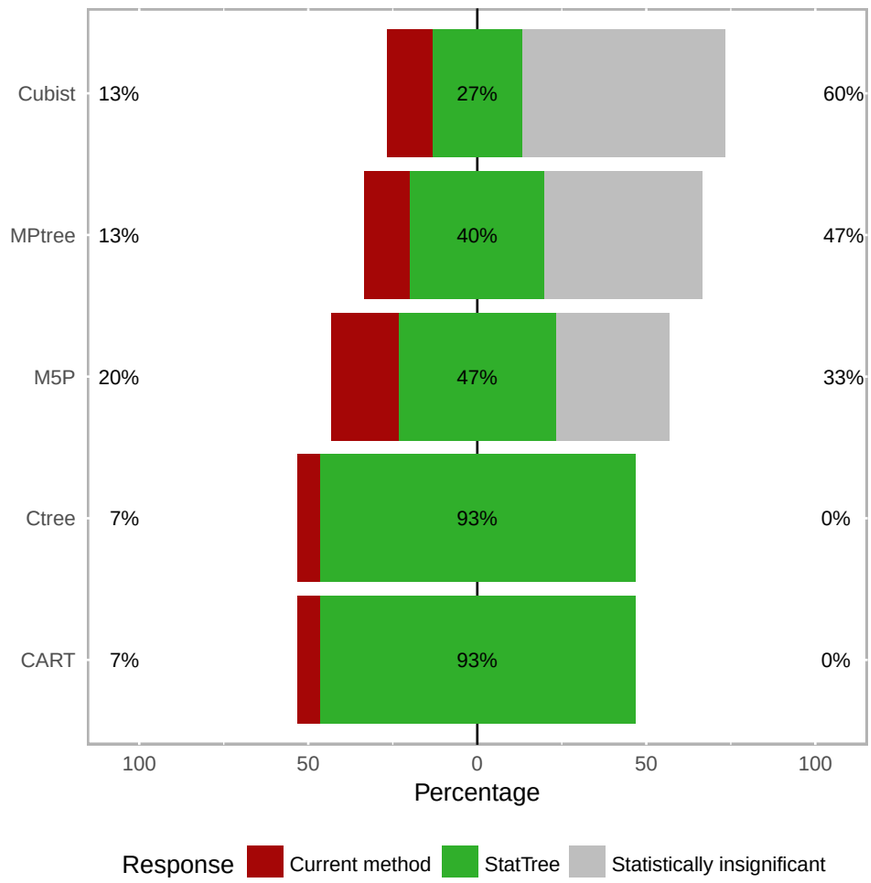


Figure 4: Percentage of winning between `StatTree` and the other methods, based only on the datasets with meaningful statistical difference.

Figure 4 is a plot of the percentage of the datasets for which `StatTree` is able to outperform the other examined approaches, considering only the examples for which the t-test null hypothesis was rejected. For example when compared to `MPtree`, 7 out of 15 examples have statistically insignificant different MAE values ($7/15 = 47\%$), `StatTree` provides better results only for 6 of the statistically significant examples ($6/15 = 40\%$) and worse results for the other 2 examples ($2/15 = 13\%$).

4.2 Variant II: Multivariate decision splittings

According to the results of section 4.1, `StatTree` has good performance against other established algorithms. In this section, the algorithm is compared to the $\texttt{StatTree}_{\texttt{LF}}$ variant. Since the optimisation model has to find the optimal coefficients for each input variable as well as the break value, it is expected to be more computationally expensive.

The results in this section test the accuracy as well as the efficiency of the optimisation model of the $\texttt{StatTree}_{\texttt{LF}}$ variant against the main `StatTree` algorithm. The validation approach followed in this section is slightly different than the rest of the work. One key difference is the analysis of the computational efficiency of the proposed optimisation model.

4.2.1 Cross-validation results

3 runs of 5-fold cross-validation are performed to test predictive performance. Table 3 that follows contains the average MAE scores of the cross-validation runs for each dataset. The variant that had the lowest error is marked with bold.

Table 3: Cross-validation results between variants using MAE

	$\texttt{StatTree}_{\texttt{LF}}$	StatTree
Concrete	5.670	**4.183**
Cooling	**1.020**	1.140
Heating	0.515	**0.368**
Yacht	0.618	**0.544**
Bodyfat	**0.181**	**0.181**
Boston	2.978	**2.545**
Dee	0.343	**0.322**
Earthquake	7.768	**7.382**
Octane	0.489	**0.439**
Pharma	0.976	**0.866**
Plastic	**1.204**	1.225
Sensory	0.653	**0.618**
Wankara	1.142	**0.995**
Abalone	**1.493**	1.527
Speeding	**4.115**	4.144

Based on the results of Table 3, the variant with multivariate splittings under performs for most of the examined datasets in this work. However, for some examples the difference between the two variants is very close. In order to determine the actual impact of the multivariate decision rules, a statistical analysis needs to be performed.

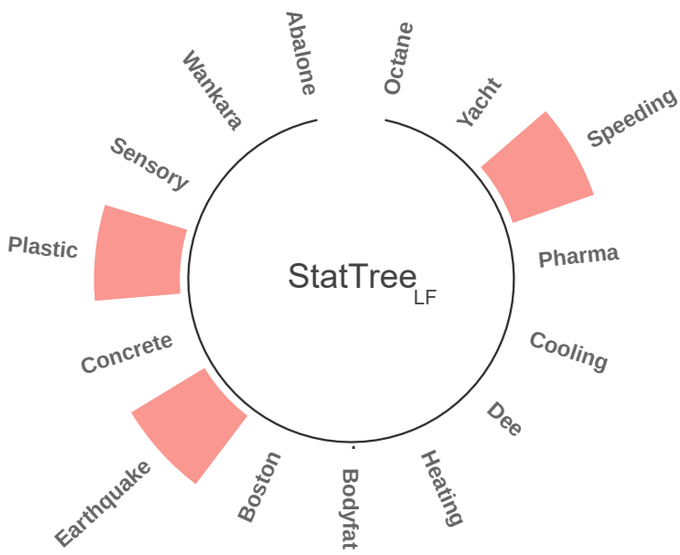
4.2.2 Statistical analysis

The statistical analysis applied in this section is similar to the one of section 4.1.2. The Welch's t-test is used to compare the results of Table 3. The two samples used for the test are the cross-validation runs for the two variants.

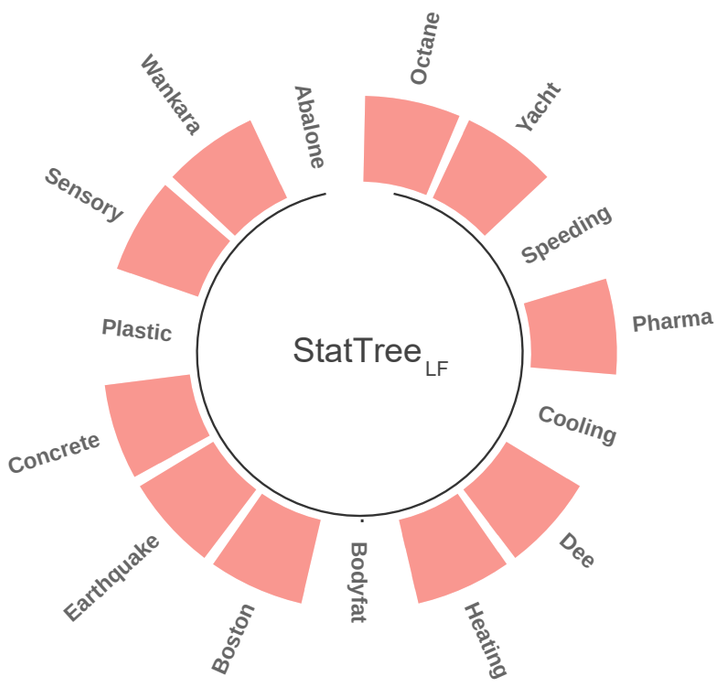
Figure 5 summarises the results of the statistical analysis. The plots in this figure are similar to the ones in the previous section, where a pairwise comparison takes place between `StatTree` and the variant with the multivariate splittings.

In Figure 5a, if a bar is present then there is a statistically significant difference in the CV results. In Figure 5b, if a bar is present then the `StatTree` variant has better MAE score compared to the $\texttt{StatTree}_{\texttt{LF}}$ variant.

Looking at both figures, it is obvious that using multivariate decision rules does not provide improved prediction accuracy for all the examples. At the same time, the results are very similar to the core algorithm since there is not enough evidence to determine if the differences are statistically significant.



(a) Welch's t-test



(b) MAE scores

Figure 5: Visualisation of the computational results of the $\texttt{StatTree}_{\texttt{LF}}$ variant. Case (a) is a representation of the statistical analysis using Welch's t-test. Case (b) is a representation of the MAE performance.

4.2.3 Analysis of computational efficiency

The final part of the comparison between `StatTree` and $\texttt{StatTree}_{\texttt{LF}}$ concerns with the computational efficiency of the two variants.

For the purposes of this test, for each one of the 3 runs of 5-fold cross-validation the average absolute gap and average computational time of solving the root node of the tree is captured. The reason for selecting to capture those metrics only for the root node of the trees, is the fact that the root node is the most computationally expensive to solve. The root node contains all the available data samples and each subsequent node in the tree contains a subset of the data, hence making the model easier to solve.

Table 4: Comparison of computational efficiency between the two variants

	$\texttt{StatTree}_{\texttt{LF}}$		`StatTree`	
	Gap	CPU (s)	Gap	CPU (s)
Concrete	1.0	200.0	0.0	31.5
Cooling	1.0	200.0	0.0	9.2
Heating	1.0	200.0	0.0	11.2
Yacht	0.8	200.0	0.0	2.3
Bodyfat	1.0	200.0	0.0	22.6
Boston	1.0	200.0	0.0	47.7
Dee	1.0	200.0	0.0	5.6
Earthquake	1.0	200.0	0.0	11.8
Octane	0.4	200.0	0.0	0.9
Pharma	0.0	99.2	0.0	1.2
Plastic	1.0	200.0	0.0	9.0
Sensory	1.0	200.0	0.0	9.1
Wankara	1.0	200.0	0.0	91.1
Abalone	1.0	200.0	0.0	202.3
Speeding	1.0	200.0	0.0	38.1

Table 4 shows the difference in computational efficiency between `StatTree` and $\texttt{StatTree}_{\texttt{LF}}$. After performing the 3 runs of 5-fold validation, it is clear that solving the optimisation model for the root node of the multivariate case, is very expensive. The model always reaches the pre-defined limit of 200s while failing to close the gap, when using the `CPLEX` solver. On the other hand, the univariate algorithm successfully closes the gap in all of the examined datasets while requiring significantly less time to do so.

It is important to note that solving the root node in the case of $\texttt{StatTree}_{\texttt{LF}}$ requires only a single `MILP` model. However, in the univariate case the algorithm has to solve an `MILP` model for every single input variable. This is why abalone requires more than 200s to solve the root node, without violating the 200s limit per model.

In order to better understand the difference in computational efficiency between the two variants, the parallel plot of Figure 6 is produced.

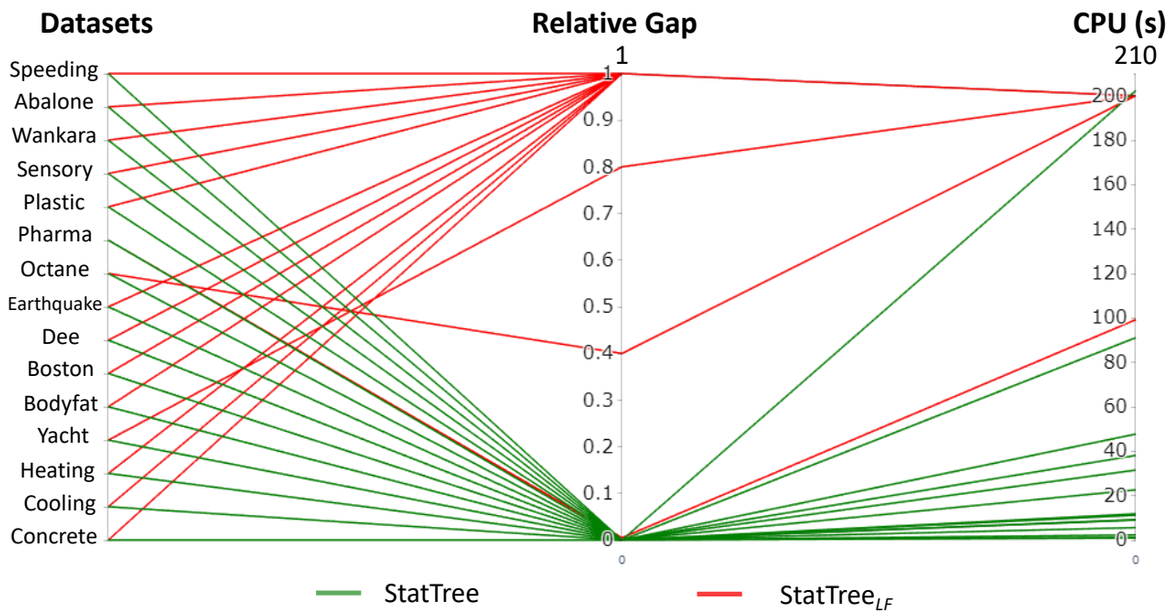


Figure 6: Comparison of computational efficiency between the two variants

In this plot, the red lines represent the multivariate variant whereas the green ones the univariate variant. It is evident from this figure, that the multivariate model struggles to close the gap and requires a long time to solve a single *MILP* model and create the decision rule.

The univariate variant is the exact opposite. It can successfully close the gap for every examined dataset and can split a node (which requires multiple *MILP* models) within an acceptable time limit.

It can therefore be concluded that the $\texttt{StatTree}_{\texttt{LF}}$ variant is not as efficient as the main `StatTree` variant, despite having similar performance for the datasets examined in this work (as demonstrated in Figure 5).

4.3 Variable selection model

The results of section 4.2.3 suggest that the `StatTree` variant is able to generate decision rules within an acceptable time limit. However, it is evident from Figure 6 that when the algorithm is presented with complex datasets (for example high dimensionality), it requires more computational time to converge to a solution.

This section compares the predictive performance of `StatTree` with and without the variable selection model that was described in Section 2.3. Constraint 12 controls the maximum number of variables to be selected by the model. At each iteration of the `StatTree` algorithm, the variable selection model is applied first in order to select a subset of the original input space to be considered for splitting. The goal is to reduce the computational time of training, without sacrificing predictive performance.

By following the same validation procedure of section 4.1.2, the variant with the variable selection model will be compared to the original `StatTree` with exhaustive search, for the same datasets and based on the same k-fold splittings. The reported results are the averages of the k-fold cross-validation runs. Once again, the same statistical analysis is performed to check for statistical significance in the results.

Table 5, contains the results of the cross-validation runs. In this work, two values were selected for the N parameter which controls the maximum number of selected variables, the extreme case of

$N = 1$ and the value of $N = 3$ which has been found to be a good fit for the examined examples.

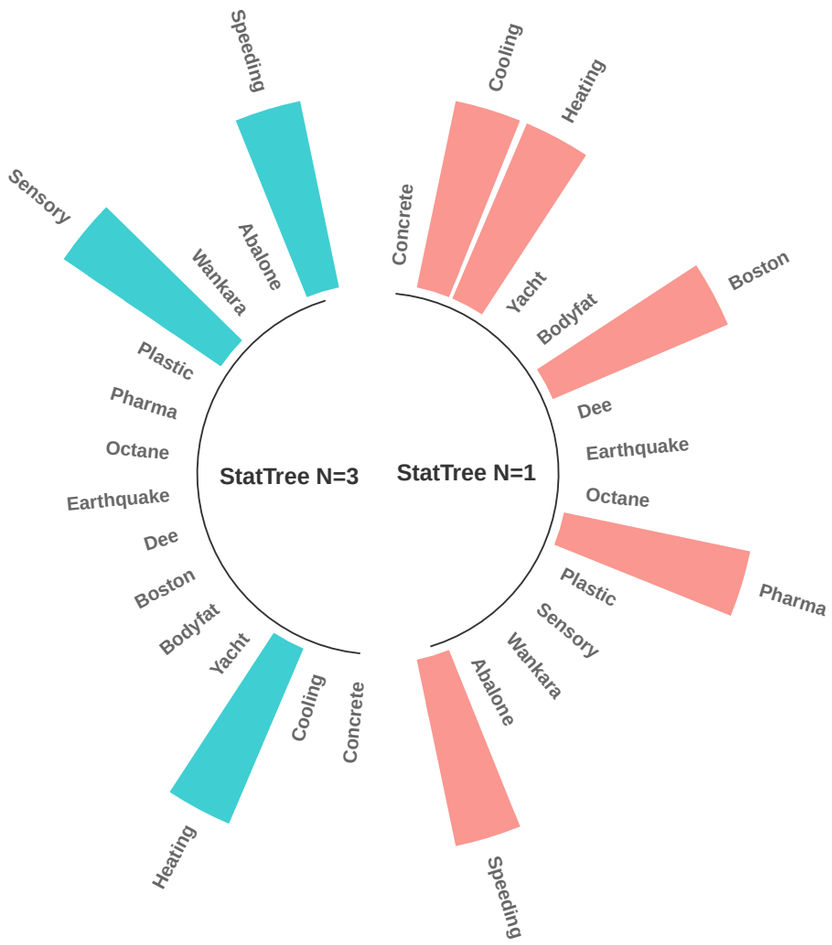
The metrics used for this comparison are the MAE score and the overall CPU time needed to complete the cross-validation runs.

Table 5: Comparison between exhaustive search and subset selection

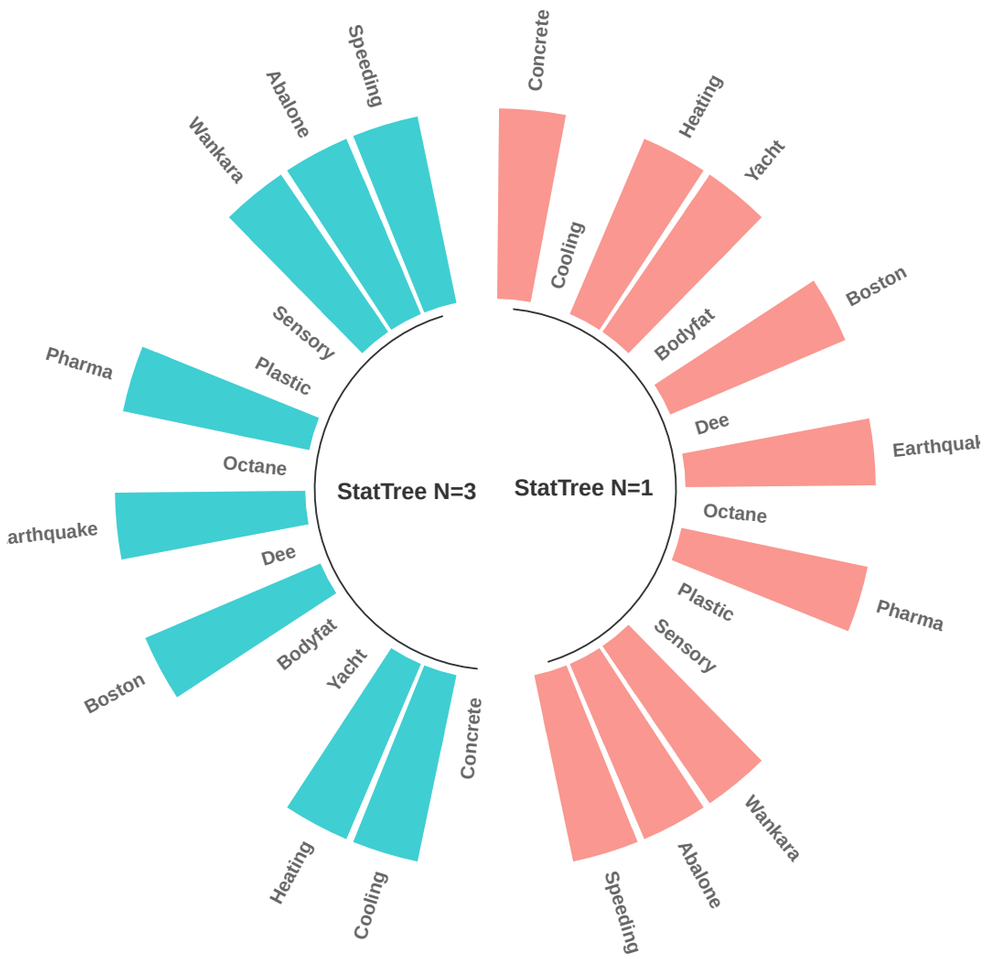
	`StatTree` - Exhaustive Search		`StatTree` with `FSelect` N=3		`StatTree` with `FSelect` N=1	
	MAE	CPU(s)	MAE	CPU(s)	MAE	CPU(s)
Concrete	4.329	618	**4.067**	469	4.728	302
Cooling	**1.175**	243	1.187	107	1.448	46
Heating	**0.367**	233	0.396	154	0.575	74
Yacht	0.539	97	0.538	43	0.540	30
Bodyfat	**0.183**	341	**0.183**	20	**0.183**	16
Boston	**2.568**	574	3.016	190	3.174	124
Dee	**0.313**	61	**0.313**	31	**0.313**	31
Earthquake	**7.345**	153	7.376	94	7.367	70
Octane	**0.391**	4	**0.391**	4	**0.391**	4
Pharma	**0.900**	16	0.905	16	1.046	12
Plastic	**1.226**	55	**1.226**	55	**1.226**	42
Sensory	0.610	81	**0.600**	18	0.609	13
Wankara	**0.972**	1215	0.987	600	1.016	758
Abalone	**1.490**	2567	1.530	1320	1.521	640
Speeding	**4.143**	700	4.357	378	4.319	460

On a first glance it is obvious that performing an exhaustive search on the entire set of input variables provides better results, but it comes at the cost of *CPU* time. Completing all the cross-validation runs requires more time when scanning through every single variable, which is expected. However, in many cases, reducing the number of variables to be considered for splitting leads to very similar predictive performance.

A statistical analysis of the results is vital in order to derive any useful piece of information. The Welch's t-test will be used, comparing the exhaustive search version of `StatTree`, with `StatTree` when $N = 1$ and $N = 3$. Figure 7 is a visualisation of that comparison. The graphs in the figure work exactly the same way as in Figure 3. It is evident that for the majority of the examined examples, there is no statistically significant difference between the full set of variables and the reduced ones.



(a) Welch's t-test



(b) MAE scores

Figure 7: Visualisation of the computational results of the variant with the `FSelect` model. Case (a) is a representation of the statistical analysis using Welch's t-test. Case (b) is a representation of the MAE performance.

5 Concluding remarks

This work addressed the issue of multivariate regression analysis by generating tree structures. The proposed method used an optimization model to split data into nodes and the Chow statistical test to control the tree generation process. The algorithm generated tree structures by deciding the partitioning variable for every node through an iterative approach and optimised the corresponding break point values and regression parameters, while minimising the summation of the absolute deviation between predictions and observed values.

Several real-world examples were used in this work in order to test the algorithm. Its performance is compared to other established tree regression algorithms that are available in the literature. Computational experiments indicated that the proposed method consistently performed well and provided competitive performance against the examined algorithms. Focusing more on the comparison against `MPtree`, which shares the same optimisation model for splitting nodes, it was seen that there was a big gain in predictive accuracy in favour of the novel algorithm. This could be an indication that the introduction of the Chow statistical test was having an impact on the generated trees. Overall, Figure 3 was a summary of this works' findings and a visual representation of predictive performance as well as the statistical analysis that was performed to check for statistical significance.

In addition to the core algorithm, two extra variants were also proposed. The first variant, called $\texttt{StatTree}_{\texttt{LF}}$, used the concept of multivariate decision rules. Instead of splitting nodes on specific input variables, the optimisation model created liner expressions as rules. These expressions were functions of the entire input space. A cross-validation analysis was performed to compare the performance of this variant against the main `StatTree` algorithm. The results indicated that the new model that was based on multivariate rules is not able to outperform the classical approach. In fact, based on the statistical analysis that was performed it can be concluded that there is little to no difference in predictive performance. However, there was a significant difference in the efficiency between the two variants. Figure 6 captures this difference, which suggests that the multivariate model struggles to converge to a solution and requires all of the 200s that had been chosen as a time limit.

The final variant is a novel mathematical model that was introduced to perform subset selection and handle the task of reducing the dimensionality of the input space when searching for the optimal partitioning variable. This novel formulation is an *MILP* model that applied a linear regression model to the data and used binary variables in order to select features. The selection was based on the minimisation of the *BIC* metric. Computational runs proved that one advantage of using this novel model was the reduced training time, which allowed the algorithm to handle datasets with a larger number of variables. Furthermore, Figure 7 illustrated that there was not a big compromise in predictive performance, since the results of Table 5 were very similar and in most cases the differences between them proved to be insignificant.

Overall, `StatTree` was deemed to be a good and competitive alternative in the realm of decision tree algorithms. Furthermore, the inclusion of the `FSelect` model was a useful addition that reduced training times without a significant sacrifice to performance. Finally, future work could be done in order to improve the efficiency of the $\texttt{StatTree}_{\texttt{LF}}$ variant, which would enable the algorithm to create multivariate decision rules and make the optimisation model usable.

Appendix A Mathematical formulation of MPtree

In this section, the mathematical programming model that was used in the `MPtree` algorithm is described as formulated by Yang et al. (2017) in the literature. This model is responsible for partitioning a node into two child nodes based on a particular partitioning feature. The mathematical model is presented is as follows:

Indices

c	child node of the current parent node; $c = left$ represents left child node, and $c = right$ represents right child node
m	feature/independent input variable
m^*	partitioning feature
n	current node
s	data samples, $s = 1, 2, ..., S$

Sets

C_n	set of child nodes of the current parent node n
S_n	set of samples in the current parent node n

Parameters

a_{sm}	numeric value of sample s on feature m
y_s	output value of sample s
u_1, u_2	suitably large positive numbers
ϵ	suitably small number

Continuous variables

B_c	intercept of regression function in child node c
D_s	training error between predicted output and real output for sample s
Pr_{sc}	predicted output for sample s in child node c
$W1_{mc}$	regression coefficient for feature m in child node c
$W2_{mc}$	regression coefficient for feature m in child node c
X_{m^*}	break-point on partitioning feature m^*

Binary variables

F_{sc}	1 if sample s falls into child node c; 0 otherwise

Mathematical Constraints

In order to assign samples into the child nodes, binary variables are introduced to the model in the following constraints:

$$a_{sm} \leq X_m + u_1 \cdot (1 - F_{sc}) - \epsilon \qquad \forall\ s \in S_n,\ c = left,\ m = m^* \tag{18}$$

$$X_m - u_1 \cdot (1 - F_{sc}) + \epsilon \leq a_{sm} \qquad \forall\ s \in S_n,\ c = right,\ m = m^* \tag{19}$$

The following constraint forces each sample to belong only to one child node:

$$\sum_{c \in C_n} F_{sc} = 1 \qquad \forall s \in S_n \tag{20}$$

For each child node c, polynomial functions of order 2 are employed to predict the value of samples (P_{sc}):

$$P_{sc} = \sum_m a_{sm}^2 \cdot W2_{mc} + \sum_m a_{sm} \cdot W1_{mc} + B_c \qquad \forall\ s \in S_n,\ c \in C_n \tag{21}$$

For any sample s, its training error is equal to the absolute deviation between the real output and the predicted output for the child node c where it belongs to and can be expressed with the following two equations:

$$D_s \geq y_s - P_{sc} - u_2 \cdot (1 - F_{sc}) \qquad \forall\ s \in S_n,\ c \in C_n \tag{22}$$

$$D_s \geq P_{sc} - y_s - u_2 \cdot (1 - F_{sc}) \qquad \forall\ s \in S_n,\ c \in C_n \tag{23}$$

The objective function is to minimise the sum of absolute training errors of splitting the current node n into its child nodes:

$$\min \sum_{s \in S_n} D_s \tag{24}$$

The resulting model can be summarised as:
objective function (24)
subject to (18)-(23) constraints
and is formulated as an MILP problem that can be solved to optimality.

Appendix B Welch's t-test

The t-test is formulated as (Ruxton, 2006):

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}} \tag{25}$$

where

$\bar{X}_1, \bar{X}_2$	mean of the 1^{st} and 2^{nd} sample respectively
s_1^2, s_2^2	variance of the 1^{st} and 2^{nd} sample respectively
N_1, N_2	size of the 1^{st} and 2^{nd} sample respectively

The degrees of freedom associated with this variance estimate is approximated as (Ruxton, 2006):

$$\nu \approx \frac{\left(\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}\right)^2}{\frac{s_1^4}{N_1^2 \cdot \nu_1} + \frac{s_2^4}{N_2^2 \cdot \nu_2}} \tag{26}$$

where

$v_1 = N_1 - 1$	degrees of freedom associated with the 1^{st} variance
$v_2 = N_2 - 1$	degrees of freedom associated with the 2^{nd} variance

Once the t-statistic and the degrees of freedom have been computed, the t distibution can be used to test the null hypothesis using a two-tailed test.

Acknowledgments

Funding from the UK Leverhulme Trust under grant number RPG-2015-240 is gratefully acknowledged.

References

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., and Herrera, F. (2011). KEEL Data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17.

Bertsimas, D. and Dunn, J. (2017). Optimal classification trees. *Machine Learning*, 106:1039–1082.

Bertsimas, D. and Shioda, R. (2007). Classification and regression via integer optimization. *Operations Research*, 55:252–271.

Bischl, B., Lang, M., Bossek, J., Horn, D., Richter, J., and Surmann, D. (2017). BBmisc: Miscellaneous Helper Functions for B. Bischl. *Available at* https://cran.r-project.org/package=BBmisc.

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

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and regression trees*. Taylor & Francis.

Brodley, C. E. and Utgoff, P. E. (1995). Multivariate decision trees. *Machine learning*, 19:45–77.

Burnham, K. P. and Anderson, D. R. (2003). *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer Science & Business Media.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20:273–297.

Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13:21–27.

Cozad, A., Sahinidis, N. V., and Miller, D. C. (2014). Learning surrogate models for simulation-based optimization. *AIChE Journal*, 60:2211–2227.

Dheeru, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository. [http://archive.ics.uci.edu/ml]. University of California, Irvine, School of Information and Computer Sciences.

Dougherty, C. (2011). *Introduction to econometrics*. Oxford University Press.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55:119–139.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, 19:1–67.

GAMS Development Corporation (2016). General Algebraic Modeling System (GAMS) Release 24.7.1, Washington, DC, USA.

Gkioulekas, I. and Papageorgiou, L. G. (2018). Piecewise regression through the akaike information criterion using mathematical programming. *IFAC-PapersOnLine*, 51(15):730–735.

Gkioulekas, I. and Papageorgiou, L. G. (2019). Piecewise regression analysis through information criteria using mathematical programming. *Expert Systems with Applications*, 121:362–372.

Hornik, K., Buchta, C., and Zeileis, A. (2009). Open-source machine learning: R meets weka. *Computational Statistics*, 24:225–232.

Hothorn, T., Hornik, K., and Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical statistics*, 15:651–674.

Hothorn, T. and Zeileis, A. (2015). partykit: A modular toolkit for recursive partytioning in r. *The Journal of Machine Learning Research*, 16:3905–3909.

Jian, W., Zhu, L., Xu, Z., and Chen, X. (2017). A variable selection method for soft sensor development through mixed integer quadratic programming. *Chemometrics and Intelligent Laboratory Systems*, 167:85–95.

Kotsiantis, S. B. (2013). Decision trees: a recent overview. *Artificial Intelligence Review*, 39:261–283.

Kuhn, M. (2008). Building predictive models in r using the caret package. *Journal of Statistical Software*, 28:1–26.

Kuhn, M. and Quinlan, R. (2017). Cubist: Rule- and instance-based regression modeling. *Available at* https://cran.r-project.org/package=Cubist. R package.

Miyashiro, R. and Takano, Y. (2015). Subset selection by mallows' cp: A mixed integer programming approach. *Expert Systems with Applications*, 42:325–331.

Muller, A. C. and Guido, S. (2016). *Introduction to machine learning with python: A guide for data scientists*. O' Reilly Media, Inc.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Quinlan, J. R. et al. (1992). Learning with continuous classes. In *5th Australian joint conference on artificial intelligence*, volume 92, pages 343–348. World Scientific.

R Development Core Team (2016). *R: A Language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria.

Ruxton, G. D. (2006). The unequal variance t-test is an underused alternative to Student's t-test and the Mann–Whitney U test. *Behavioral Ecology*, 17:688–690.

Sok, H. K., Ooi, M. P.-L., and Kuang, Y. C. (2015). Sparse alternating decision tree. *Pattern Recognition Letters*, 60:57–64.

Therneau, T., Atkinson, B., and Ripley, B. (2018). Package rpart. *Available at* https://cran.r-project.org/package=rpart.

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

Verwer, S. and Zhang, Y. (2017). Learning decision trees with flexible constraints and objectives using integer optimization. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 94–103. Springer.

Vlachos, P. (2005). StatLib-statistical datasets. *Available at* http://lib.stat.cmu.edu/datasets/.

Wagenmakers, E.-J. and Farrell, S. (2004). AIC model selection using Akaike weights. *Psychonomic Bulletin & Review*, 11:192–196.

Wang, F., Wang, Q., Nie, F., Li, Z., Yu, W., and Ren, F. (2020). A linear multivariate binary decision tree classifier based on k-means splitting. *Pattern Recognition*, page 107521.

Wang, F., Wang, Q., Nie, F., Yu, W., and Wang, R. (2018). Efficient tree classifiers for large scale datasets. *Neurocomputing*, 284:70–79.

Wang, Y. and Witten, I. H. (1996). Induction of model trees for predicting continuous classes.

Welch, B. L. (1947). The generalization of 'student's' problem when several different population variances are involved. *Biometrika*, 34:28–35.

Wilson, Z. T. and Sahinidis, N. V. (2017). The alamo approach to machine learning. *Computers & Chemical Engineering*, 106:785–795.

Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Yang, L., Liu, S., Tsoka, S., and Papageorgiou, L. G. (2016). Mathematical programming for piecewise linear regression analysis. *Expert Systems with Applications*, 44:156–167.

Yang, L., Liu, S., Tsoka, S., and Papageorgiou, L. G. (2017). A regression tree approach using mathematical programming. *Expert Systems with Applications*, 78:347–357.

Yeh, I. C. (1998). Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research*, 28:1797 – 1808.