UNIVERSITY COLLEGE LONDON

DOCTORAL THESIS

Optimisation based approaches for machine learning

Author:	Supervisor:
Ioannis Gkioulekas	Prof. Lazaros Papageorgiou

A thesis submitted in fulfillment of the requirements for the degree of *Doctor of Philosophy*

in the

Centre for Process Systems Engineering
Department of Chemical Engineering

November, 2020

Declaration of Authorship

I, Ioannis Gkioulekas, declare that this thesis titled, “Optimisation based approaches for machine learning” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

Abstract

Machine learning has attracted a lot of attention in recent years and it has become an integral part of many commercial and research projects, with a wide range of applications. With current developments in technology, more data is generated and stored than ever before. Identifying patterns, trends and anomalies in these datasets and summarising them with simple quantitative models is a vital task. This thesis focuses on the development of machine learning algorithms based on mathematical programming for datasets that are relatively small in size.

The first topic of this doctoral thesis is piecewise regression, where a dataset is partitioned into multiple regions and a regression model is fitted to each one. This work uses an existing algorithm from the literature and extends the mathematical formulation in order to include information criteria. The inclusion of such criteria targets to deal with overfitting, which is a common problem in supervised learning tasks, by finding a balance between predictive performance and model complexity. The improvement in overall performance is demonstrated by testing and comparing the proposed method with various algorithms from the literature on various regression datasets.

Extending the topic of regression, a decision tree regressor is also proposed. Decision trees are powerful and easy to understand structures that can be used both for regression and classification. In this work, an optimisation model is used for the binary splitting of nodes. A statistical test is introduced to check whether the partitioning of nodes is statistically meaningful and as a result control the tree generation process. Additionally, a novel mathematical formulation is proposed to perform feature selection and ultimately identify the appropriate variable to be selected for the splitting of nodes. The performance of the proposed algorithm is once again compared with a number of literature algorithms and it is shown that the introduction of the variable selection model is useful for reducing the training time of the algorithm without major sacrifices in performance.

Lastly, a novel decision tree classifier is proposed. This algorithm is based on a mathematical formulation that identifies the optimal splitting variable and break value, applies a linear transformation to the data and then assigns them to a class while minimising the number of misclassified samples. The introduction of the linear transformation step reduces the dimensionality of the examined dataset down to a single variable, aiding the classification accuracy of the algorithm for more complex datasets. Popular classifiers from the literature have been used to compare the accuracy of the proposed algorithm on both synthetic and publicly available classification datasets.

Impact Statement

A number of novel machine learning algorithms are proposed in this thesis to handle regression and classification tasks. The aim of this project is the development of such algorithms under a mathematical programming and mixed integer programming framework. All of the algorithms are based on the principle of segmenting the data into smaller subsets and fitting a model to each individual partition. Emphasis is placed on model interpretability and ease of use.

This project was motivated by the increasing interest in machine learning and data science not only from the tech industry, but other fields as well. In addition to algorithm development, another goal for this project is to address the need for algorithms that are easy to understand for scientists with little or no knowledge of machine learning, such as researchers from Chemical Engineering and Process Systems Engineering. Considering this goal, the development of the algorithms has been done in such a way that requires minimum input from the user in order to tune the parameters of the algorithms. The resulting models are highly interpretable.

Several impact areas are expected to benefit from this research. In bioinformatics, segmented regression has been used for the development of Quantitative Structure-Activity Relationship (*QSAR*) models. The novel proposed algorithm could provide new insights, as it improves upon the work of the current literature of piecewise methodologies. Furthermore, decision trees have been extensively used in many domains, including engineering. Energy consumption, predictive maintenance and fault diagnosis are a few examples. Applications beyond engineering include the financial sector, fraudulent transaction detection, supply chain and more. Creating additional novel algorithms to add to the current literature, can help extend the knowledge of these fields.

In order to put the performance of the methods in this thesis into a perspective, all of the results generated in this work have been validated and compared against established algorithms from the literature. This has been achieved using state-of-the-art open source libraries that are available in popular programming languages.

Acknowledgements

Firstly, I would like to express my gratitude to my supervisors Professor Lazaros Papageorgiou and Dr. Michail Stamatakis for helping me complete this project and all the useful and critical discussions. Their guidance and trust throughout my PhD were invaluable. I would also like to thank Professor Lazaros Papageorgiou and Professor David Bogle for giving me the opportunity to be a teaching assistant during my studies and further increase my interest in optimisation and teaching.

Financial support from the UK Leverhulme Trust is gratefully acknowledged.

I am also very thankful for all my colleagues at UCL who have provided a lot of experience and help. Special thanks to Jude Ejeh, for all the help and support while working together as teaching assistants. I would also like to extend my gratitude to Professor Vivek Dua and Dr. Sergio Medina-Gonzalez for sharing their expertise and insights during our collaboration.

Many thanks also go to my friends from Greece, Foteini, Markos, Thanasis, Lana and Niki as well as to my friends from the Product and Process Systems Engineering office of the Chemical Engineering Department at UCL. I thank Harry, Marco, Alba, Panos, Sergio, Andres and Arun for their friendship and support.

I would like to say special thanks to my friends and flatmates Andreas and Dimitris for their support and all the good times we had that really helped me through my studies.

I am extremely grateful to my family for supporting and believing in me. Their support was a great motivation to keep pushing with my work. Thank you to my parents Alkiviadis and Paraskevi, my brother Konstantinos and my beloved nephew Alkiviadis and niece Ermioni.

Contents

List of Figures

List of Tables

1 Introduction

This thesis is concerned with the development of novel supervised learning algorithms for regression and classification. The focus is the development of mixed-integer optimisation models to address the learning tasks. In this chapter, the domain of machine learning is introduced, along with the major types of learning and their respective challenges. The main goals of this work are summarised.

We currently live in a "data era" where vast amounts of data are created and collected every day. Due to this growth in the availability of data, machine learning and data science have gained increased popularity over the last decades due to the variety of applications, even to simple everyday tasks. Machine learning involves the use of scientific processes and methods to analyse and draw conclusions from data and it is an agglomeration of several fields including computer science, statistics, mathematical optimisation and more (Grus, 2019).

This domain has infiltrated our daily lives with countless applications such as (Mohri et al., 2012):

- Text or document classification , e.g. spam detection
- Natural language processing
- Speech recognition
- Computer vision tasks, e.g. image recognition and face detection
- Fraud detection e.g. credit card payments
- Social networks
- Recommendation systems and search engines

and many others.

Machine learning and artificial intelligence are at the centre of what is known as *Industry 4.0*, which is the automation of current traditional industries using modern smart technologies (dos Santos et al., 2018).

Advances in computer technology have enabled the storage and processing of large amounts of data, as well as accessing it remotely through the internet. Cloud computing, advanced analytics and big data have largely impacted many industries, with chemical engineering being one of them.

Artificial intelligence and machine learning systems have been applied to chemical engineering processes to help identify patterns in data that would take significantly longer for a human to find (Cartwright, 2020).

Machine learning methodologies have been applied to various areas of chemical engineering, with examples including modelling and simulation (Chetouani et al., 2007; Khayet and Cojocaru, 2012), process control (Shah and Gopal, 2016; Chaffart and Ricardez-Sandoval, 2018) and process optimisation (Medina-González et al., 2020; Petsagkourakis et al., 2020) to name a few.

Outside of commercial applications, machine learning has been extensively applied for research purposes, allowing many fields to adopt data-driven approaches.

However, information that might be hidden in the data becomes useful only when it is analysed and translated in such a way that it can be easily used in the future. The application of machine learning methods to databases is often called *data mining*, where the objective is to construct mathematical models that should have the ability to learn from the data and make reliable predictions for future applications (Witten et al., 2016).

Thanks to the increased amount of interest, a lot of research has been done in the domain of machine learning and the development of algorithms that differ in their approach, the type and size of data they can handle as well as the task they are solving.

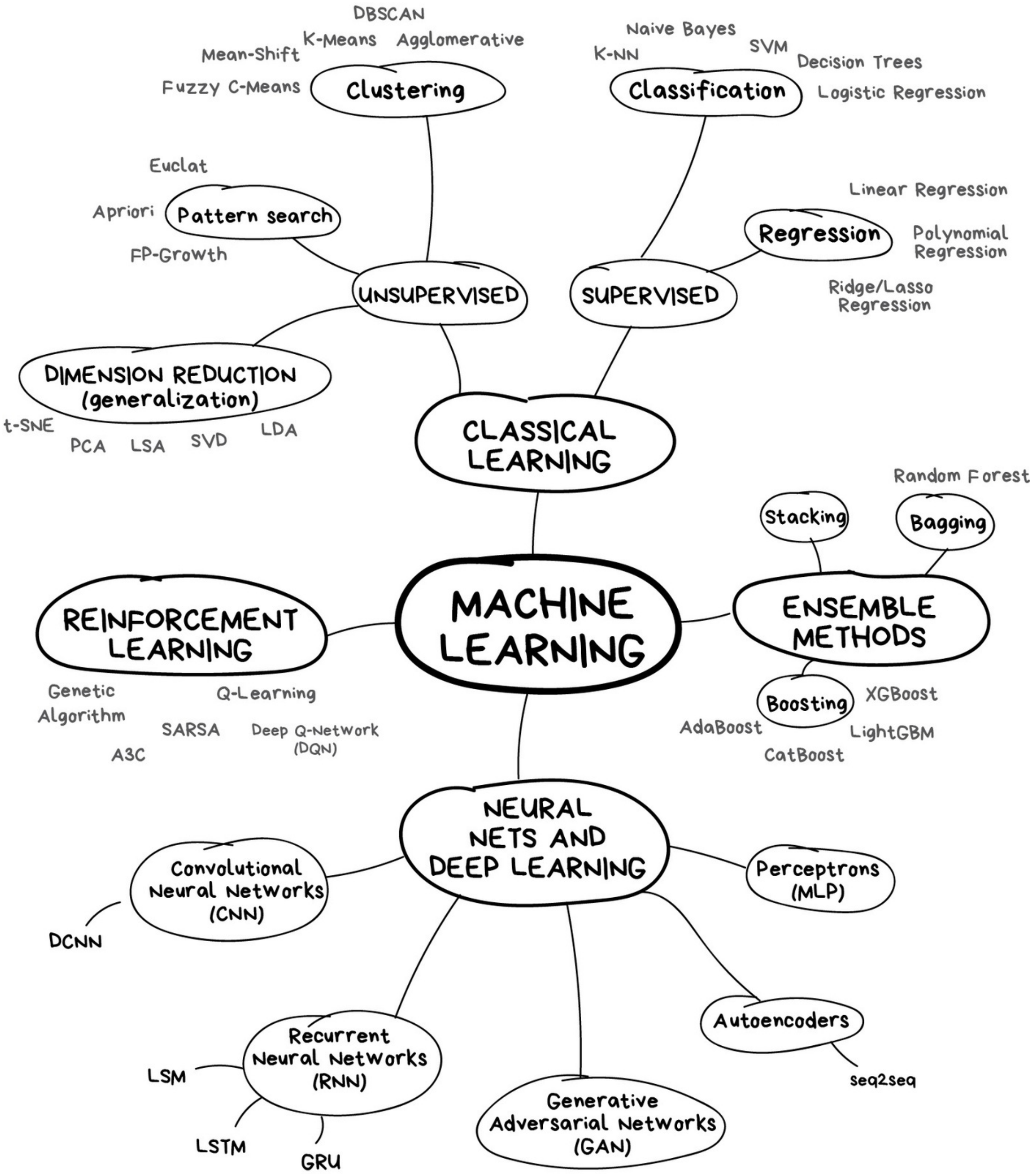


FIGURE 1.1: The machine learning world (Vasily Zubarev, 2018).

As a result of the diversity of the ongoing research, this field is larger than ever before, with various types of learning in existence, such as supervised and unsupervised learning, reinforcement learning and deep learning to name a few. Figure 1.1 is a visual overview of the domain of machine learning.

This doctoral thesis aims to develop machine learning algorithms based on mathematical programming and optimisation techniques to deal with supervised learning cases and datasets that are relatively small in size.

1.1 Supervised learning

In supervised learning, given a set of data $D = \{(x_n, y_n), n = 1, ..., N\}$ the task is to learn the relationship between the input x and output y such that, when given a novel input x^* the predicted output y^* is accurate (Barber, 2012).

The term "supervised" indicates that there is a notional "supervisor" specifying the output y for each input x in the available data D. Upon receiving such a set of labelled training samples, the learner captures this mapping of input and output and can make predictions for new unseen points. This supervised learning approach is the one associated with tasks such as classification and regression.

1.1.1 Classification and regression

Two major types of supervised learning problems include classification and regression. Variables can be characterised as either quantitative or qualitative. Quantitative variables can take numerical values such as a person's age or the average temperature of a city. On the other hand, qualitative variables take on values that belong in one of K different categories, also known classes. Problems that can handle quantitative output variables are referred to as regression problems whereas problems that can handle categorical output variables are referred to as classification problems (James et al., 2013).

Specifically, classification is a modelling technique that assigns new samples to known groups. The goal is to create a set of mathematical rules that separate samples into known classes; these are choices from a predefined list of possibilities.

Meanwhile, regression is a modelling technique that creates a set of mathematical rules that capture the relationship between dependent and independent variables. The goal of regression is to produce a continuous number that predicts the dependent variable or output as accurately as possible.

In supervised learning, constructing an accurate predictive model involves fitting it to a set of training data and then fine-tuning its parameters in such a way that this model will be able to make reliable predictions on new untrained data.

1.1.2 Challenges in supervised learning

Overfitting is a common concern when constructing any predictive machine learning model. Creating a supervised learning model includes fitting it to a set of training data. During this process, if the mathematical functions of the model fit the data points too closely, then the resulting model will overfit future predictions. This usually means that the selected model is too complex and is easily affected by all the peculiarities that exist in the data. Consequently, this model has poor predictive performance (Hawkins, 2004). On the other end, there is underfitting, which results from a model that is too simple and therefore unable to produce good predictions.

This problem of over and under fitting is also known as the *bias vs variance* tradeoff. As mentioned, creating a model that will be able to generalise well to new data is a challenging task. Figure 1.2 below is a standard example of demonstrating the difference between bias and variance.

Imagine playing a game of darts. The goal is to hit the target at the centre of the board. If we are able to consistently hit the target, then there is low bias and variance in our accuracy. However, if all of our shots are all concentrated but away from our target, then it is obvious that for some reason our accuracy is biased to hit that specific spot on the board.

On the other hand, if our shots are scattered across the board with no apparent pattern, then there is variance in our aim. In this case, some of our shots might actually hit the target, but it is obvious that overall we are not very good.

Extending this example to statistics and machine learning, the bias error derives from probable bad assumptions in the learning process, resulting in a model that has not captured the underlying relationship of the data. This is normally associated with underfitting.

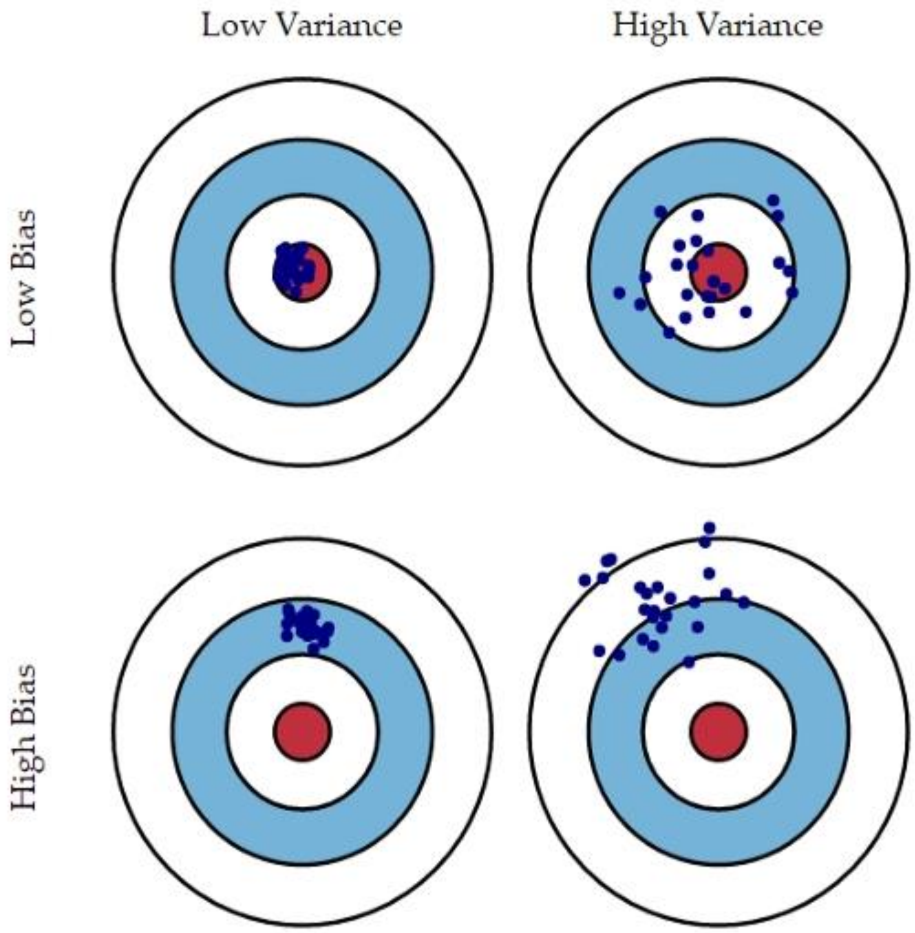


FIGURE 1.2: An example of bias vs variance (Fortmann-Roe, 2012).

In contrast, a high-variance model is one that is very sensitive and is affected by small fluctuations in the data. This is an indication that the model is so tailored to the specific training set that it is also describing the data's noise.

In machine learning, this problem is about creating a model that will have good predictive performance and it will be able to generalise well to new data. To test the performance of a model, it is a fairly standard approach to split the data for training and testing purposes, a process which is explained in detail in Section 2.4, and use various metrics to quantify performance.

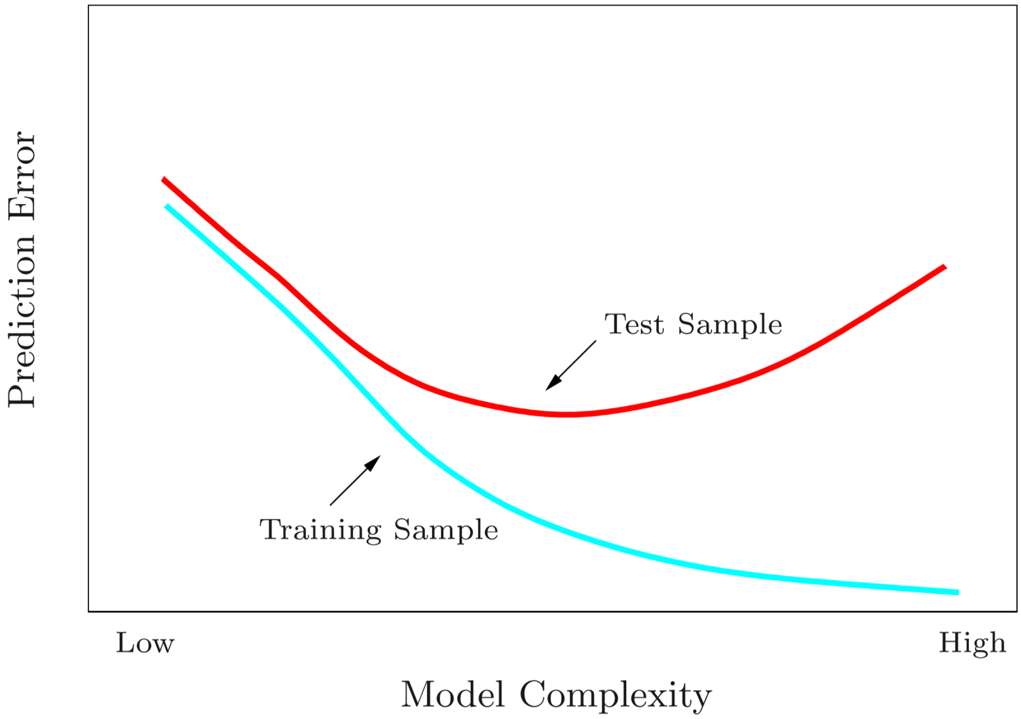


FIGURE 1.3: Test and training error as a function of model complexity (Hastie et al., 2008)

Figure 1.3 illustrates the impact that model complexity has on performance. Low model complexity can lead to a model with high bias meaning that there is a large prediction error in both training and testing. High model complexity can lead to a model with high variance resulting in a big performance gap between the training and testing phase.

1.2 Unsupervised learning

In unsupervised learning, given a set of data $D = \{x^n, n = 1, ..., N\}$ the aim is to find a compact description of the data (Barber, 2012). The learner this time receives a set of unlabelled training data points and makes decisions for new unseen data. Unsupervised tasks can be challenging since there is no known output variable to map and predict.

Two major types of unsupervised learning include dimensionality reduction and clustering algorithms.

1.2.1 Clustering

Clustering, as the name suggests, takes unlabelled data and finds subgroups, also known as clusters. Clustering algorithms seek to partition the data and create clusters in such a way that the samples within the same clusters are similar, whereas the samples between different clusters are less similar.

This ‘similarity‘ is expressed mathematically by explicitly defining an appropriate metric. This metric is often a domain-specific consideration that is made based on knowledge of the examined data (Witten et al., 2016).

1.2.2 Dimensionality reduction

Dimensionality reduction is the process of reducing the size of high-dimensional data consisting of a large number of input variables to find a more compact representation, while retaining as much of the original data characteristics as possible. This process is important as it can explore the correlation between the variables and eliminate the highly correlated ones. Such a process is also useful for saving both memory and computational time (Muller and Guido, 2016).

1.2.3 Challenges in unsupervised learning

In contrast to the challenges of supervised learning, unsupervised learning is different and quite challenging. Since there is no target variable or output, there is no good way of knowing whether the unsupervised task has performed "well".

It is very hard to assess the obtained results and there is no universally accepted mechanism for performing validation on an independent data set (James et al., 2013). This lack of a validation method often leads to the necessity of manual inspection to validate the results. This is why it is common for unsupervised techniques to be used as part of an exploratory data analysis (Muller and Guido, 2016).

1.3 Motivation and contribution of the thesis

In recent years, there has been a great increase in the computational power of Mixed Integer Optimisation (*MIO*) solvers such as GUROBI (Gurobi Optimization, LLC, 2019) and CPLEX (IBM, 2019). These improved solvers are capable of handling problems of considerable size. Bixby (2012) tested a set of *MIO* problems on the same computer using various versions of the CPLEX solver from 1991 through 2007. The speedup factor was measured to be more than 29,000. Combined with major improvements in computer hardware, researchers are able to tackle a selection of statistical and machine learning problems using integer optimisation.

Optimisation techniques using mixed integer programming as well as machine learning methodologies, have been applied to solve chemical engineering problems in the past. However, the focus in this work is placed on algorithm development. These novel algorithms use mixed integer programming techniques and expand upon the existing literature of machine learning methods. In this section, the motivation of this thesis is discussed as well as the motivation for developing novel regression and classification algorithms that are interpretable and transparent.

1.3.1 Development of a piecewise regression algorithm

The first part of the thesis addresses the development of a piecewise regression algorithm. Piecewise regression methods using linear expressions have the advantage of simplicity and model interpretability due to the use of linear models, but identifying the position of the break points is a challenging task. Piecewise approaches use break points to partition the input data into segments. Those break points are the boundaries between the different segments. As a result, the final regression model consists of a number of segments, also known as regions, their corresponding break point values and an explicit regression model that is fitted to each region.

Piecewise regression models have been used in the past for various applications. Toms and Lesperance (2003) used piecewise regression in order to model abrupt changes in ecological data. Specifically, this work used data from a study which looked at understory plant communities along a transition from clearcut to old-growth forest. The objective of the analysis was to locate a threshold that represented a difference in plant composition. Hence, piecewise regression provided those thresholds as the break points of the model.

Another application of piecewise regression was presented by Malash and El-Khaiary (2010). This research study demonstrated that piecewise linear regression can be used as a tool to analyse experimental absorption data. Previous mechanistic models suffered from uncertainties that were caused by the different absorption rates for different time periods, and it was usually up to the researcher to visually examine and decide those segments. The piecewise regression approach used in this work on the other hand, provided good estimates.

In a more recent study, Muggeo et al. (2020) used data from the European Centre for Disease Prevention and Control (*ECDC*) for COVID-19. The authors used piecewise regression to quantify the effect of the lockdown on some well-known measures which are typically measured to monitor an epidemic progression.

In terms of available software, the segmented package, which is available for the `R` programming language, enables users to fit segemented models to data. However, the user has to specify the number of regions as well as initial estimates for the position of the break points (Muggeo, 2008, 2003). This software has some important limitations though. The first one is the fact that it is difficult to reasonably supply good starting points especially for multivariate datasets, where data visualisation is impossible. The second limitation is that only the partitioning feature can have different regression coefficients across the segments, while the other input variables keep the same coefficients across the whole range.

Many research on piecewise regression has been application driven, while software and algorithm availability is quite limited. This issue was partially adressed by Yang et al. (2016), with the development `OPLRA` (Optimal Piecewise Linear Regression Analysis). This algorithm is a piecewise regression approach that splits data into regions and fits a linear regression model to each one. One of the main drawbacks of this algorithm is the selection of the optimal number of regions. The authors proposed a heuristic rule which is based on an iterative approach. A series of *Mixed Integer Linear Programming* (*MILP*) models are solved and with each iteration a new region is added to the model. The termination criterion uses a parameter introduced by the authors. This parameter is used as a threshold for the reduction of the error between iterations. This parameter is fixed to a specific value based on the findings of a sensitivity analysis.

However, fixing this parameter can lead to overfitting when the algorithm is presented with new datasets. This work proposes an extension of that method that will address this issue and decide the final number of regions with a more robust and established way. The new algorithm uses information criteria to deal with the model selection task and improve prediction accuracy.

Four different variants are proposed that can be split into two main approaches. Iterative and non-iterative approaches. The iterative ones use the mathematical model of `OPLRA` for splitting data into regions and post-process the results to use the information criteria. The non-iterative approaches extend the mathematical formulation and integrate the criteria into the optimisation. These approaches can identify the number of regions without the need of iterations.

1.3.2 Development of a decision tree regressor

The next part of this thesis focuses on generating tree regression models. In previous work, Yang et al. (2017) developed a tree regression algorithm, called `MPtree`, that employed an *MILP* mathematical formulation to optimise the value of the break points when splitting nodes. Generating a tree structure involves solving that *MILP* model recursively, splitting nodes into two sub-nodes until a stopping criterion is satisfied.

However, there are two major drawbacks with that algorithm. The first one is the stopping criterion, which uses a heuristic rule that includes a parameter that the authors defined based on a sensitivity analysis. This fixed value results in poor performance when new examples are introduced to the algorithm. Hence, there is a need for a better solution.

The second drawback concerns the practical use of the algorithm. Since the nature of tree models requires partitioning the data repeatedly, the `MPtree` algorithm becomes impractical when it is used to fit datasets with a large number of variables. In this case, the problem is that for each node the algorithm performs an exhaustive search through the entire input space, solving an *MILP* model for each input variable separately until the best one is identified. To deal with this issue, a novel mathematical formulation is proposed that will be used as an embedded subset selection model to enable the algorithm to search for the optimal partitioning variable on a reduced set of input variables.

1.3.3 Development of a decision tree classifier

This work focuses on the development of a decision tree classifier. Decision trees are easy to understand and powerful for making predictions. Current classification tree approaches partition the input data recursively into nodes and generate tree structures with the goal of creating homogeneous tree branches, containing samples of specific classes (Breiman et al., 1984). Standard tree classifiers assign a specific class membership to leaf nodes. As a result they compromise part of the accuracy of the classification in the case of small trees, or overfit the data in the case of large trees.

In this work, a novel tree classifier is proposed with the aim of addressing such issues. The classifier uses an optimisation model to split nodes into subsets and identify the optimal partitioning variable. The mathematical formulation of this algorithm also fits a linear expression to the data at each splitting, creating a new pseudo-feature based on which the classification task is performed. The classification is achieved by partitioning this new feature into multiple regions, also referred to as "ranges" in this work, and assigning a single class membership to each range. This mathematical formulation is applied recursively in order to generate a tree structure.

The resulting trees contain breaking rules similar to those of other established classifiers. However, each leaf node can represent multiple classes at once, since each node contains multiple ranges that are based on the linear transformation of the mathematical model. This approach can create more compact tree structures that are easy to understand.

2 Literature review

In this chapter, a deeper analysis of classical machine learning is given focusing on the differences of supervised and unsupervised learning. Additionally, some of the most popular algorithms for each respective learning type are described and the basics of model selection are also presented. This review aims to give an overview of the current state in machine learning.

2.1 Supervised learning

In supervised learning, the task is to fit a function to the data that will map the input-output relationship that exists in the data. This mathematical function is created by a set of labeled training data. Two major categories with a plethora of applications of both industrial and academic interest, are regression and classification.

2.1.1 Classification and regression

Many algorithms exist in the literature that handle such tasks with various degrees of complexity. Algorithms that are more complex are more demanding in computational resources and often harder to interpret. On the other hand, less complex approaches are quick to compute and easier to understand. Such is the case of linear regression, which models the relationship of variables using linear expressions. Logistic regression is also a simplistic approach, which despite its name, is used for binary classification tasks.

Other more sophisticated approaches include *Multivariate Adaptive Regression Splines* (MARS) (Friedman, 1991), *Support Vector Machines* (SVM) (Cortes and Vapnik, 1995) and *K-Nearest Neighbors* (KNN) (Cover and Hart, 1967).

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) and a piecewise regression approach called *Optimal Piecewise Linear Regression Analysis* (OPLRA) (Yang et al., 2016).

Decision tree 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. One popular metric being used for the creation of these splits is the information gain (Quinlan, 1986), which is based on information entropy. Information entropy can be viewed as the amount of information that there is in an event (Shannon, 2001). The more certain or deterministic an event is, the less information it contains. Information gain is based on the difference of entropy when a split occurs to the data.

Popular tree algorithms include *Classification and Regression Trees* (CART) (Breiman et al., 1984), *Iterative Dichotomiser 3* (ID3) (Quinlan, 1986), C4.5 (Quinlan, 1993), *Conditional Inference Trees*, (CTree) (Hothorn et al., 2006), M5P (Quinlan et al., 1992; Wang and Witten, 1996) and Cubist.

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 has been developed (Verwer and Zhang, 2017), that uses integer programming to construct trees of certain depth from data sets of sizes up to 1000 samples. 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. The key difference between this approach and other classic methods is the creation of decision rules based on multivariate splits.

Ensemble methods have gained a lot of interest due to their good predictive ability. Those 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).

Another category that has attracted attention is boosting. Boosting is very similar to ensemble methods, 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 better predictive performance than the original. Popular methods include *Adaptive Boosting* (`AdaBosot`) (Freund and Schapire, 1997) and `XGBoost` (Chen and Guestrin, 2016).

2.1.2 Description of literature algorithms

Linear Regression

In linear regression, the relationship between input and output variables is modelled using linear functions. If $X^T = (X_1, X_2, ..., X_m)$ is the vector of input variables, then the linear model takes the following form (Hastie et al., 2008):

$$Y = \beta_0 + \sum_{j=1}^{m} X_j \cdot \beta_j$$

where:

β_0	regression intercept
β_j	regression coefficients
m	input variables

The unknown model parameters can be estimated from the known values of the data. One of the most popular estimation methods is least squares, where the objective is to minimise the residual sum of squares.

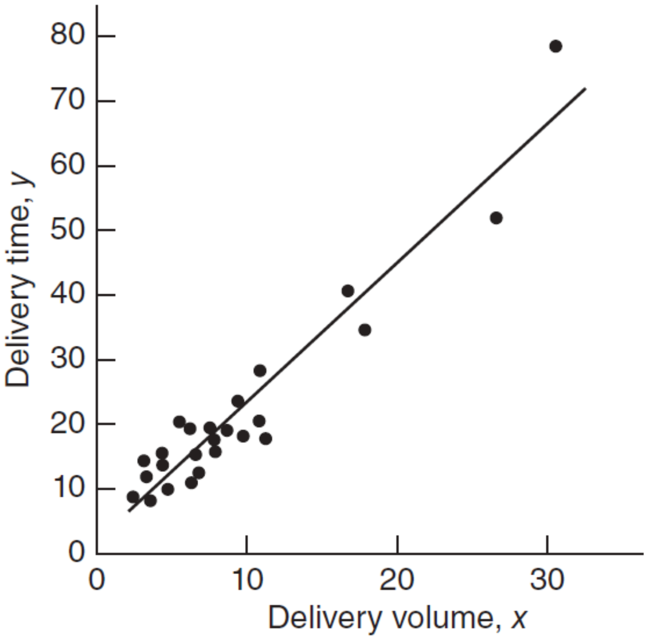


FIGURE 2.1: Straight-line relationship between delivery time and delivery volume (Montgomery et al., 2012)

Figure 2.1 is an example of estimating the relationship between delivery time and delivery volume of a product for a soft drink beverage company. The data points suggest a correlation between time and volume; in fact the impression is that the data points fall along a straight line (Montgomery et al., 2012).

Multivariate Adaptive Regression Splines - MARS

MARS is a regression approach that can be seen as an extension of linear models and is well suited for high-dimensional problems. MARS creates models that have the following form (Friedman, 1991):

$$Y = \sum_i c_i \cdot B_i(x)$$

The model is a weighted sum of basis functions $B_i(x)$, with c_i being constant coefficients. The basis functions that are used in MARS are hinge functions which take the form of:

$$max(0, x - t) \quad \text{and} \quad max(0, t - x)$$

The building process of MARS includes two steps, the forward and backward steps. In the forward step, the algorithm starts from a single intercept term and then adds pairs of hinge functions in order to minimise training error.

At the end of the process, the final model can become very large and this could potentially overfit the data. Hence, the backward step is an elimination step where the terms whose removal leads to the smallest increase in residual squared error are deleted (Hastie et al., 2008).

K-Nearest Neighbors - KNN

K-nearest neighbors is an algorithm that can be used for classification and regression problems. In KNN, given a positive integer K and a test observation, the algorithm identifies the K points (known as neighbors) in the training data that are closest to the testing observation. For regression problems, the outcome is a single value that is the average of all the values of those K nearest points (James et al., 2013). KNN is very quick to compute since there is no training phase to construct a model.

Support Vector Machines - SVM

Support vector machines is a widely used learning algorithm that is based on the concept of decision planes that define decision boundaries. A decision plane is one that separates a set of objects having different class memberships. The goal is to find a plane that has the maximum margin between the available classes.

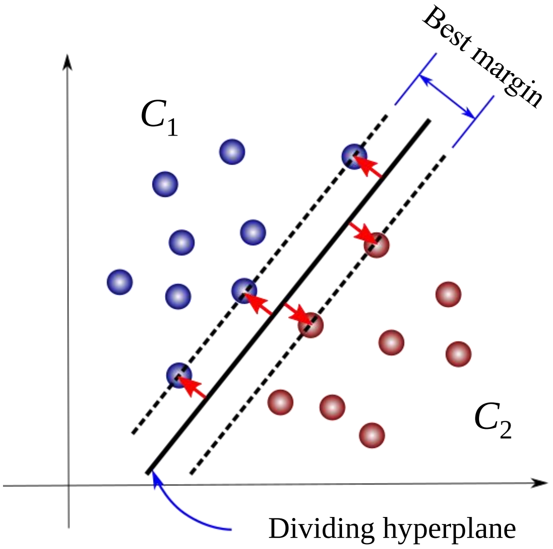


FIGURE 2.2: Visual representation of SVM (Carrasco, 2019).

However, the algorithm can also be used for regression analysis. The goal is to find a function that deviates from the response by a value no greater than a threshold, called ϵ. Basically, the method minimises two terms in the objective function, one of which is the ϵ-insensitive loss function and the other is the model complexity (Hastie et al., 2008).

One important feature with support vector machines is the use of kernels. Kernels can map the dataset from the original space to a higher dimensional space using a set of mathematical functions. By doing this process, the mapped objects can become linearly separable.

Decision Trees

Decision tree learning is a method that is popular in the machine learning domain, with various algorithms present in the literature. `CART` is an algorithm that creates binary trees. As mentioned earlier, tree models involve selecting input variables and splitting them into two subsets recursively, until a tree has been constructed. The selection of a specific input variable and breaking point is achieved through the use of a greedy algorithm where different split points are tested using a cost function. The split with the minimum cost is selected.

The recursive splitting procedure is stopped when a criterion is satisfied during the training phase. A common criterion is the minimum number of samples in a leaf node. The smaller the number, the larger the final constructed tree will be.

The final step is pruning. Simple trees are preferable since they are easy to understand and less likely to cause overfitting issues. Pruning techniques assess the quality of leaf nodes and remove the ones that offer no improvement in the overall performance, hence creating a more compact tree representation (Breiman et al., 1984).

Decision trees do have some advantages over other supervised learning approaches such as (James et al., 2013):

- Easy and quick to compute
- Easy to understand and explain
- Can be graphically represented, especially if they are small

However, decision tree learners can create over-complex structures that overfit the training data.

Figure 2.3 is an example of a decision tree classifier. This example concerns the development of a method for identifying high risk heart attack patients, once they have been admitted to the hospital, on the basis of the initial 24-hour data. This structure classifies incoming patients as *HR* or *LR* depending the yes-no answers to at most three questions (Breiman et al., 1984).

As mentioned in Section 2.1.1, another popular decision tree method is `CTree`. This algorithm tackles the problem of recursive partitioning in a statistical framework (Hothorn et al., 2006). For each node, the association between each independent input feature and the output variable is quantified using a permutation test and multiple testing correction. If the strongest association passes a statistical threshold, a binary split is performed in that corresponding input variable; otherwise the current node is a terminal node.

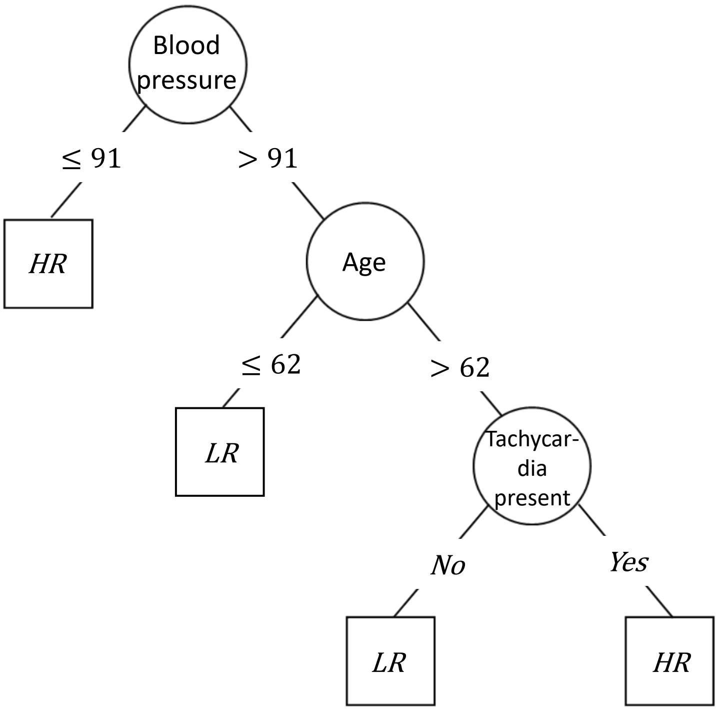


FIGURE 2.3: An example of a decision tree structure (Breiman et al., 1984).

M5P can be considered as an improvement of CART. The tree generation process is the same, but instead of having values at the leaves, the constructed trees can have multivariate linear models; these tree models are analogous to piecewise linear functions (Quinlan et al., 1992). In the first stage, the algorithm generates the tree and instead of using information gain for the binary splits, it uses a different criterion. According to the authors, the splitting criterion that is used minimises the intra-subset variation in the class values down each branch. In the second stage, pruning is applied to the tree back from each leaf. The difference of this step compared to the one in CART is that when pruning to an interior node, consideration is given to replace that node by a regression plane instead of a constant value (Wang and Witten, 1996).

Cubist (Rulequest, 2020), which is a commercially available rule-based regression model, employs M5P to grow a tree first, which is then collapsed into a smaller set of if-then rules. This is achieved by removing and combining paths from the root to the terminal nodes. One important thing to note it the ability of the model to have overlapping `if-then` rules. In other words, samples can be assigned to multiple rules. A final value is produced by averaging all the predictions.

Ensemble methods

Ensemble methods are algorithms based on the hypothesis that combining multiple machine learning models into a single predictive model can improve results (Hastie et al., 2008). Those multiple models are often called "weak learners" and each one is trained to solve the same problem. In the end, those models are combined to obtain better results. Ensemble methods can be divided into two major groups:

1. Sequential methods, where the base or weak learners are generated sequentially, and the overall performance can be boosted by weighing previously mislabelled examples with higher weights.
2. Parallel methods, where the base learners are generated in parallel. The basic motivation of parallel methods is to exploit the independence between the weak learners and reduce the overall error by averaging.

In parallel methods, the most popular approach for training the 'weak' independent learners is called *bagging* (which stands for bootstrap aggregating). Bagging randomly creates T subsets by taking samples with replacement from the original dataset. Then, T models are trained based on those subsets and the final results are aggregated to achieve better predictive performance. For aggregating the results of the base learners, bagging uses voting for classification and averaging for regression (Muller and Guido, 2016).

One well-established algorithm that uses bagging is *Random Forest*, which uses decision tree algorithms as base learners and generates multiple models in parallel to improve performance.

The difference with sequential methods is that the various combined weak learners are no longer trained independently. Each model in the sequence focuses on the weaknesses of the previous iteration by trying to improve prediction accuracy on the samples that were badly handled. At the end of the process, a strong learner with lower bias is obtained (Breiman, 1996).

Multi-layer perceptron - MLP

`MLP` is a type of feedforward neural network. It consists of an input layer, a number of intermediate layers that are known as hidden layers and an output layer. Each layer has a number of nodes, also known as neurons. The network is fully connected, which means that the neurons in each layer are connected to all the neurons in the two neighbour layers. With the exception of the input layer, the rest of the neurons use an activation function which defines the output value of those neurons (Hastie et al., 2008). The use of appropriate activation functions, such as sigmoid or *ReLU* (Rectified Linear Unit), allows `MLP` to model data with non-linear relationship (Gevrey et al., 2003). In terms of training the network, backpropagation is employed.

2.2 Unsupervised learning

Unsupervised learning techniques are quite different, due to the lack of an output variable. As mentioned in Section 1.2, unsupervised techniques can be useful for performing exploratory data analysis and extracting useful information or creating different representations of the available data. Popular methodologies include clustering, dimensionality reduction and various data transformation techniques.

2.2.1 Clustering

Clustering is one of the most important unsupervised learning domains with many tools in existence. However, each clustering algorithm has its own strengths and weaknesses due to the complexity of available information. Traditional algorithms can be divided into various categories, with 4 of the most popular adhering to the following principles (Xu and Tian, 2015):

- partitioning
- hierarchy
- density
- distribution

Clustering algorithms that are based on partitioning, organise the data into clusters and regard the centre of the data points as the centre of the corresponding cluster. These algorithms are efficient but can be sensitive to initial conditions and outliers. Popular algorithms include `k-means` (MacQueen et al., 1967) and `k-medoids`.

Hierarchy-based algorithms work differently. They partition the instances in either a top-down or bottom-up way. Agglomerative clustering assigns every instance to its own cluster and then successively merges neighbouring clusters (bottom-up approach). On the other hand, divisive clustering works in the reverse order. All the instances belong to a single cluster and then they are successively divided into sub-cluster (top-down approach) (Maimon and Rokach, 2005).

Density-based methods identify distinctive clusters in the data, based on the idea that a cluster in the data space is a continuous region of high point density, separated from other such clusters by continuous regions of low point density. By design, these algorithms do not assign outliers to clusters since those data points do not fall into any regions of high density (Kriegel et al., 2011). Popular algorithms include `DBSCAN` (Ester et al., 1996) and *Mean-shift* (Comaniciu and Meer, 2002).

Finally, distribution-based approaches assume that the samples are composed of distributions. Samples that are generated from the same distribution belong to the same cluster, otherwise they belong to different clusters (if there exist several distributions in the original data).

2.2.2 Dimensionality reduction

In cases where the input dataset has a large number of variables, it is often helpful to reduce its dimensionality or to find a lower dimensional representation.

Some of the benefits of reducing the dimensionality of the data include:

- Reduced computational expenses, since the new reduced dataset can help speed up all the subsequent operations.
- Ability to visualise the data for exploratory analysis.

Approaches can be divided into feature selection and feature extraction (Mohri et al., 2012). Feature selection methods retain only a subset of the original input variables and eliminate the rest from the model. There are a number of different types of feature selection methods such as wrapper, filter and embedded methods (Hastie et al., 2008).

Wrapper methods search the input space to create subsets of features and evaluate each subset by fitting a model on the subset. The methods can be computationally expensive. Similarly, filter algorithms also create subsets of features but the evaluation is based on a proxy metric, which is easy and quick to compute instead of fitting a model. On the other hand, embedded techniques are model-specific, since the creation of a subset is part of the model construction process.

Feature extraction transforms the data from a high-dimensional to a low-dimensional space, compressing the data and potentially finding a representation that is more informative for further processing. One of the most widely used algorithms for feature extraction is *Principal Component Analysis* (PCA) (Hotelling, 1933). Another popular algorithm is *autoencoder* (Kramer, 1991).

Principal Component Analysis

PCA is a technique for reducing the dimension of a dataset and create new features, called principal components. This technique uses an orthogonal transformation to convert a set of correlated variables into a set of uncorrelated ones (James et al., 2013). The first principal component is created by being placed at the direction that maximises the variance of the data points. The next principal component is perpendicular to the first one. It is worth noting that in a high-dimensional case, each subsequent principal component is placed along an axis that maximises the variance of the data, subject to the constraint that is orthogonal to the preceding component (Witten et al., 2016).

The user can specify the number of desired components, which is usually based on the cumulative captured variance of the original data, and create a reduced representation of the data. One of the most common applications of PCA is visualising high-dimensional datasets, by selected either a 2D or 3D representation (2 or 3 principal components respectively) (Muller and Guido, 2016).

2.3 Data transformations

Transformations are very useful since these algorithms create new representations of the original input space hence making the data easier to understand and handle, hence enabling the extraction of useful information. It is very common before training a supervised learning model, to pre-process the data through various transformations in order to normalise the range of independent variables.

This is very useful because in many applications, the range of raw data values can vary and lead to specific variables being more dominant than others in the final model (Hastie et al., 2008). Two of the most common normalisation techniques are feature scaling through **min-max** normalisation and **standardisation** (also known as **Z-score** normalisation).

Min-Max feature scaling

This type of normalisation shifts the data such that all the input variables are within a specified range. It is common for this range to be [0,1]. In this instance, the data values are calculated by the following formula:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2.1}$$

where x is the value of a data point for a specific variable and x_{min}, x_{max} are the maximum and minimum values of that variable respectively.

In the general case where the user wants to specify a different range, the formula can be adjusted to a custom range of [a,b] as follows:

$$x' = a + \frac{x - x_{min}}{x_{max} - x_{min}} \cdot (b - a) \tag{2.2}$$

where a and b are the range values specified by the user.

Standardisation

Feature standardisation shifts the value of the input variables in the data to have zero-mean and unit-variance. This method is commonly used for machine learning algorithms such as support vector machines and artificial neural networks (Grus, 2019). This transformation is given by the following formula:

$$x' = \frac{x - \bar{x}}{\sigma} \tag{2.3}$$

where x is the original feature vector, $\bar{x}$ is the mean of that feature vector and σ is the standard deviation.

2.4 Model selection and evaluation

According to Figure 1.3, model complexity has an impact on performance. Deciding on the final complexity of a model requires trial and error until the final model has been chosen. During this process, every time a parameter is changed, a new model is constructed. So in the end, there exists a set of candidate models to choose from and the objective is to determine which one best approximates the data.

Standard methodologies for these types of problems include cross validation and model selection criteria such as information criteria.

2.4.1 Information criteria

Information criteria are measures of the relative goodness of fit of a statistical model. Two of the most popular are the *Akaike Information Criterion* (*AIC*) and the *Bayesian Information Criterion* (*BIC*).

The *AIC* and *BIC* have been established as two of the most frequently used criteria in the literature for model selection problems with a wide variety of applications. A couple of examples include the wine industry (Snipes and Taylor, 2014), where different models are compared in an attempt to explore the relationship between ratings and prices of wines, and cancer research, where the *AIC* was used to develop a prognostic model in patients with germ cell tumors who experienced treatment failure with chemotherapy (International Prognostic Factors Study Group, 2010)

These two criteria have also been used in a number of statistical and machine learning methods such as outlier detection (Lehmann and Lösler, 2016) and feature selection. Kimura and Waki (2018) proposed a branch-and-bound search algorithm formulated as a mixed integer nonlinear programming problem, minimising the value of the *AIC* to perform feature selection. Sato et al. (2016) have also used both the *AIC* and *BIC* as measures of the goodness-of-fit to perform feature selection for logistic regression models.

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

$$AIC = -2 \cdot \ln(\hat{\mathcal{L}}) + 2K$$

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

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

The *AIC* establishes a relationship between the Kullback-Leibler measure and maximum likelihood estimation method (Fabozzi et al., 2014). It is an estimate of the relative distance between the truth and the model that approximates it. The criterion is based on the idea that no model exists that perfectly describes the truth so the best we can do is approximate it. Given a set of candidate models the criterion can identify the model that performs the best.

The *BIC* however arises from a Bayesian viewpoint and belongs to a class of criteria that are ‘dimension-consistent‘ which differ from those that are estimates of the Kullback-Leibler measure. The formulation of the *BIC* is very similar to the *AIC* but the main difference is that the *BIC* is derived to provide a consistent estimator of the dimension of the data (Burnham and Anderson, 2003).

The theory of the likelihood function begins with a probability model, given its parameters (θ). Assume a model f that describes the probability distribution of a dataset **X**. What is known is the form of this model and the parameters that describe it, $f(\mathbf{X}|\theta)$.

The likelihood function associated with this probability model differs only in terms of what is known or given. In the probability model, the parameters, the model and the sample size are known. The interest lies in the probability of observing a specific event (Burnham and Anderson, 2003).

However, in much of science neither the model parameters nor the model is known. In the likelihood function the data are given or observed, and the model is assumed. This time, the interest lies in the estimation of the unknown parameters of the model, thus the likelihood is a function of only the parameters.

The notation for the likelihood can be considered as $\mathcal{L}(\theta|\mathbf{X}, g)$, which describes the likelihood of a particular numerical value of the unknown parameter θ, given the data **X** and a particular model g. The expression $ln(\mathcal{L})$ in the information criteria, is the numerical value of the log-likelihood function at its maximum point.

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

$$AIC = n \cdot \ln\left(\frac{RSS}{n}\right) + 2K \tag{2.4}$$

$$BIC = n \cdot \ln\left(\frac{RSS}{n}\right) + K \cdot \ln(n) \tag{2.5}$$

where:

RSS residual sum of squares

2.4.2 Cross-validation

Resampling methods are a fundamental tool in statistics and involve repeatedly drawing samples from a set and refitting a model. Cross-validation (CV) is a well-known resampling technique that is used to estimate the test error that is associated with a given learning method in order to evaluate its performance. Test error is the error that results from using a learning algorithm to predict the output of previously unseen observations that were not part of the training phase (James et al., 2013).

Validation set approach

One of the simplest ways to estimate the test error of fitting a model to a set of data is the validation set approach. This approach randomly reserves a certain amount of data samples for testing and uses the remainder for training. The model is fitted only on the training set and the testing set is used to provide an estimate of the expected error of the model (Muller and Guido, 2016).

Leave-one-out cross-validation

A variation of the validation set approach is the leave-one-out cross validation. This approach involves splitting the data into training and testing sets. However, the testing set is a single sample and the training test is the rest of the dataset. This procedure is repeated until every sample has been used as part of the testing set. The resulting model can have bias compared to the validation set approach since it benefits from the larger size of the training set. Despite being time consuming, it can provide good estimates on small datasets (Muller and Guido, 2016).

K-fold cross-validation

In k-fold cross-validation, the dataset is split into k equal partitions or folds, and at each iteration a single fold is used as testing while the other ones for training.

Parameter k is user defined and usually takes the value of either 5 or 10. The procedure of fitting the model on the $k - 1$ folds and evaluating the error on the remainder fold is repeated k times and the result is k estimates of the test error which are typically averaged to obtain the overall error estimate.

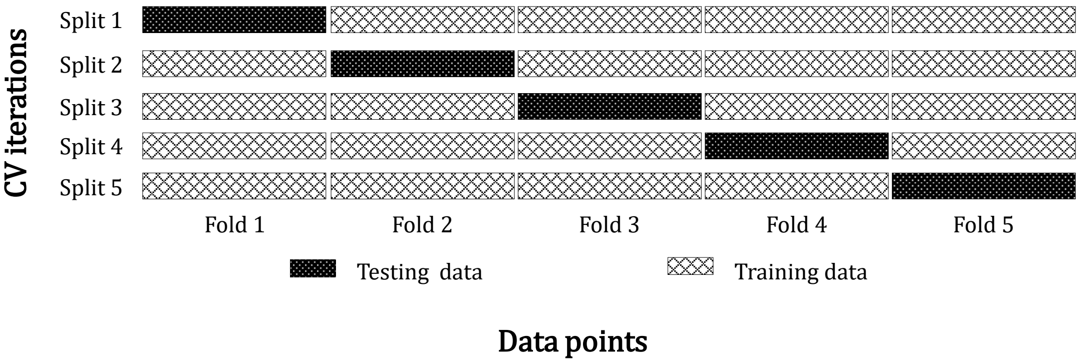


FIGURE 2.4: Data splitting in 5-fold cross-validation (Muller and Guido, 2016)

Figure 2.4 illustrates an example of 5-fold cross validation. The dataset is randomly split into 5 subsets and 5 separate models are trained, each time selecting different folds for training and testing. One obvious advantage of k-fold is computational time. In leave-one-out the training procedure is repeated n times, where n is the number of samples in the dataset. Furthermore, since the testing set consists of multiple data points, the obtained error estimate might be more accurate than simply randomly picking a single observation to evaluate the model, especially for larger datasets.

2.5 Machine learning in chemical engineering

It is important to point out how machine learning and artificial intelligence have been applied to the chemical engineering community. The following summary is not meant to be a comprehensive review, but a representative survey to demonstrate recent relevant activities.

Even though machine learning principles have been applied to chemical engineering in the past, recent advances in software and hardware has made it easier to process and handle large amounts of data in order to extract information (Venkatasubramanian, 2019). Work has also been done in the development of data-driven models to produce hybrid models using machine learning and classical chemical engineering methods (Psichogios and Ungar, 1992; Thompson and Kramer, 1994)

Many topics in chemical engineering can benefit from this interest in artificial intelligence. A few that stand out are materials design, process operations, fault diagnosis, biomedical and biochemical engineering.

Machine learning is already being used to monitor the performance of oil wells in the industry, with the goal of improving production and minimising downtime (Kellner, 2015).

In materials design, the goal is to use machine learning techniques and aid the design or discovery of materials with desired properties in a quicker and cheaper way (Venkatasubramanian et al., 1994). The design and discovery of catalysts using artificial intelligence is also a topic with considerable excitement (Medford et al., 2018; Tran and Ulissi, 2018).

Machine learning and AI-based models can have an impact on the chemical engineering community. But as important as it might be, it should always be handled with caution due to their inability to "understand" the underlying physical or chemical mechanism (Venkatasubramanian, 2019).

2.6 Machine learning and decision making under uncertainty

Many important problems involve decisions to be made under uncertainty. Accounting for the sources of unpredictable and sometimes uncontrollable conditions is a major challenge. Variations in key parameters and the data used to mathematically model a system can lead to unexpected deviation from the predicted behaviour of a system (Kochenderfer, 2015).

Many parameters involved in optimisation problems are subject to uncertainty due to a variety of reasons. Such uncertain parameters can include product demands in process planning (Liu and Sahinidis, 1996), kinetic constants in reaction-separation-recycling system design (Acevedo and Pistikopoulos, 1998) and task duration in batch process scheduling (Li and Ierapetritou, 2008) to name a few. This issue of uncertainty could render the solution of a problem infeasible.

To deal with uncertainty, a number of solution techniques have been developed, such as stochastic programming, chance constrained optimisation and robust optimisation. Nowadays however, the emergence of machine learning and the availability of tools and computational power have enabled researchers to handle decision making from a different perspective (Ning and You, 2019).

A significant amount of research has been done to incorporate machine learning and handle uncertainty. A few examples are briefly mentioned below. Ning and You (2018) proposed a data-driven robust optimisation framework that leveraged the power of principal component analysis and kernel smoothing for decision making under uncertainty. Another data-driven static robust optimisation framework was examined by Shang et al. (2017), that was based on support vector clustering and aimed to find a hypersphere with minimal volume to enclose uncertainty data. Finally, Medina-González et al. (2020) proposed a scenario reduction algorithm that combines graph theory and lasso regression to represent information as a network and identify clusters. This algorithm was applied to two-stage stochastic problems.

Although conventional techniques are the most recognised modelling paradigms for tackling uncertainty, it is apparent that data-driven mathematical programming frameworks have experienced a rapid growth due to the big interest in machine learning and is an active domain of both academic and industrial interest (Ning and You, 2019).

2.7 Concluding remarks

In this chapter, some key elements of machine learning and artificial intelligence are presented. Those key elements include supervised and unsupervised learning, description of some popular machine learning algorithms as well as model evaluation techniques and metrics. These topics cover the basics of a data science workflow, where the scientist has to manipulate some data, use an algorithm to find any patterns that exist in the data and validate the performance of the model.

This work focuses on the development of novel supervised learning algorithms. These algorithms are based on the idea of segmenting the data into smaller subsets in order to make the prediction task easier. Those novel algorithms are making use of mathematical and integer programming techniques. Previous work on algorithms that follow the principle of data segmentation (e.g. decision trees) and make use of integer programming are mentioned in this chapter. These previous studies serve as a background for the novel mathematical models developed in the current thesis.

The literature review in this chapter aims to give an idea of the level of understanding of the current state of classical machine learning, without going into much detail to the rest of the machine learning domain which covers topics such as reinforcement learning, artificial neural networks and deep learning.

3 Development of a piecewise regression algorithm

This chapter addresses the topic of piecewise linear regression. A description of a current algorithm is given, upon which two approaches are proposed and extend the mathematical formulation in order to accommodate information criteria in the optimisation. The new algorithm is tested using real-world datasets that are available through online data repositories and its performance is validated against other established algorithms from the literature.

3.1 Introduction

In linear regression analysis, the variance of the response is described using linear expressions. However, it is possible that the relationship between input and output variables are more complex. In some cases, there are data points where the value of the output changes abruptly when the input changes. Such points are called break points (Muggeo, 2003).

Piecewise regression is the procedure of identifying such break points and fit different models to each segment in order to capture the trends that are hidden within each section. Implementing a simplistic regression method like linear regression, which is easy to explain and interpret, as a piecewise approach will result in a predictive model that has better predictive capabilities and also provide insights to better understand the data.

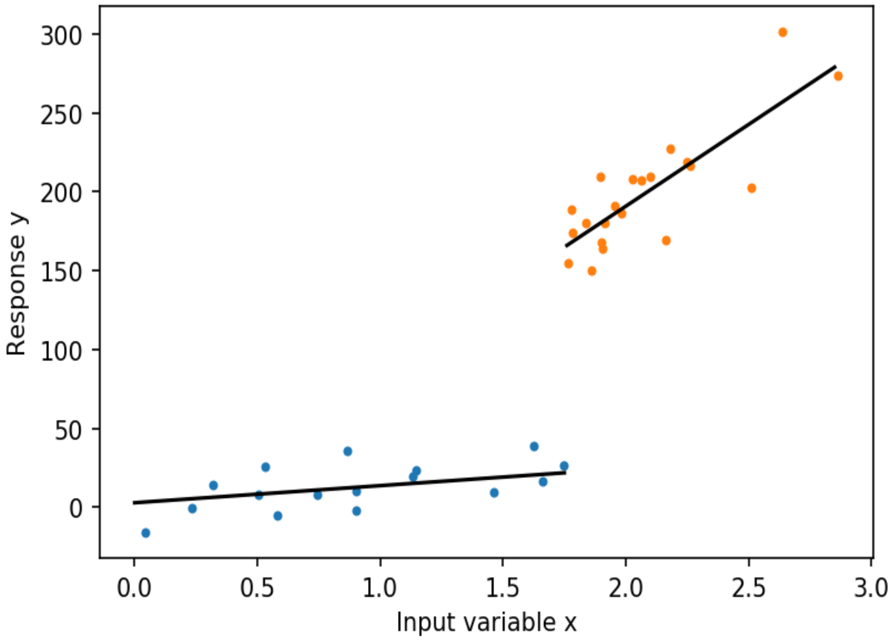


FIGURE 3.1: Illustration of a piecewise regression example with one input and one output variable

Figure 3.1 is a simplified example to demonstrate a case where there is evidence of linear correlation between input variable x and response y but there are two distinct regions. As mentioned in Section 1.3.1, identifying the number of regions and the values of the break points is a challenging task. The work in this chapter describes a series of variations of a piecewise regression algorithm that fits linear expressions to a set of data.

3.2 Proposed algorithm

This section introduces two approaches of a piecewise regression algorithm. This algorithm is using the optimisation model of the `OPLRA` algorithm (presented in Section 3.2.1). Both of the variations include information criteria in order to ensure a balance between prediction accuracy and model complexity. The key difference of the two variants is the way they utilise the information criteria.

The first approach adopts an iterative procedure for the task of selecting the optimal number of regions, by post-processing the results of the optimisation model to calculate the value of the criteria and use them to terminate the algorithm. In this case, there is no need for the user to specify any hyper-parameters before using the algorithm to fit a set of data.

On the other hand, the second approach is working on a different principle. The optimisation model is now extended to include the information criteria in the formulation. Instead of post-processing the results, the mathematical model is now capable of directly optimising the value of the criteria and determining the optimal number of regions. This is achieved by selecting an upper bound for the maximum number of allowed regions in the model. The values of the *AIC* and the *BIC* will determine the final number of regions. Sections 3.2.1, 3.2.2 and 3.2.3 that follow describe the original `OPLRA` mathematical model as well as the variants in more detail respectively.

3.2.1 Mathematical formulation of OPLRA (Yang et al., 2016)

In this section, the `OPLRA` mathematical programming model is described as formulated by Yang et al. (2016). The model accepts as input a multivariate dataset, splits it into multiple segments on a specific variable, fits a linear function to each segment, calculates the optimal regression coefficients and intercepts while minimising the summation of the absolute deviation of the fitting. All of the indices, parameters and variables that are used in the formulation are explained below:

Indices

s	data samples, $s = 1, 2, ..., S$
m	features/input variables $m = 1, 2, ..., M$
r	regions, $r = 1, 2, ..., R$
m^*	partitioning feature/variable

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

Variables

W_{mr}	regression coefficient for feature m in region r
B_r	intercept of regression in region r
Pr_{sr}	predicted output for sample s in region r
X_{mr}	break-point value of feature m for region r

Positive variables

D_s	training error between predicted output and observed output for sample s

Binary variables

F_{sr}	1 if sample s falls into region r; 0 otherwise

Mathematical Constraints

The following model works by specifying the number of regions and then splitting the data. The break point values ,X_m^r , are variables to be optimised.

So, for a given number of regions R, the first task is to arrange all of the breaking points into ascending order. For a breaking point to exist, at least two regions have to be selected. Furthermore, the number of breaking points will always be one less than the number of regions.

$$X_{m,\,r-1} \leq X_{mr} \qquad \forall\, m = m^*,\ r = 2, 3, ..., R-1 \tag{3.1}$$

Having ordered the break points, the next step is sample assignment. A new set of binary variables is introduced ,F_{sr} for the allocation of samples. For a specific region r, if a sample s belongs to that region then F_{sr} will take the value of 1. Otherwise it will be 0. The next two *bigM* constraints ensure that samples will be allocated to the correct regions.

Equation 3.2 checks the sample allocation of the region that is to the right of a specific break point, whereas Equation 3.3 checks in regards to the region that is to the left. Basically, if a sample s has a value A_{sm} that is greater than a break point value, then the sample is always to the right of that break point. Similarly, if sample s has a value less than a breaking point then it belongs to the left of that break point.

$$X_{m,\, r-1} - U_1 \cdot (1 - F_{sr}) + \epsilon \leq A_{sm} \qquad \forall\, s,\ r = 2, 3, ..., R,\ m = m^* \tag{3.2}$$

$$A_{sm} \leq X_{mr} + U_1 \cdot (1 - F_{sr}) - \epsilon \qquad \forall\, s,\ r = 1, 2, ..., R-1,\ m = m^* \tag{3.3}$$

Parameter ϵ is added to the model to make sure that no values of the dataset will equal any of the break points.

The next equation is a logical constraint that ensures that each sample s will belong only to one region r. This is achieved by forcing the summation for each sample of the binary variables F_{sr} for all regions to be equal to 1.

$$\sum_r F_{sr} = 1 \qquad \forall\, s \tag{3.4}$$

Each region has an explicit linear regression model. Equation 3.5 fits a linear regression model to each region r. W_{mr} are the slopes of the linear expressions and B_r the intercepts. Variable Pr_{sr} is the predicted response of the model.

$$Pr_{sr} = \sum_m A_{sm} \cdot W_{mr} + B_r \qquad \forall\, s, r \tag{3.5}$$

However, Equation 3.5 generates a predicted value for a sample s for each region r, despite the allocation of that sample s to a specific region. The correct prediction is ensured by Equations 3.6 and 3.7. These constraints formulate the absolute deviation between prediction and observed value for sample s, but at the same time contain *bigM* terms in order to ensure the correct allocation of samples and therefore calculate the correct deviation.

$$D_s \geq Y_s - Pr_{sr} - U_2 \cdot (1 - F_{sr}) \qquad \forall s, r \tag{3.6}$$

$$D_s \geq Pr_{sr} - Y_s - U_2 \cdot (1 - F_{sr}) \qquad \forall s, r \tag{3.7}$$

Finally, the objective function of the model is the minimisation of the absolute deviation of the error:

$$\min \sum_s D_s \tag{3.8}$$

The resulting model is formulated as an *MILP* problem.

In this literature work, the authors proposed a heuristic procedure in order to identify the partitioning variable and find the optimal number of regions. This was achieved by using an iterative approach and introducing a new parameter β, which was used as a threshold to the reduction percentage of the absolute error. If the reduction percentage of the error was above that parameter, then a new region was added and the model would be solved again. The entire process stops once convergence has been achieved.

3.2.2 Iterative approaches

Since the task at hand is to decide the optimal number of regions, the assumption can be made that each time a new region is introduced, a new regression model is created with more degrees of freedom. As a result information criteria are chosen to address the issue of model selection.

Using Equations 2.4, 2.5 and modifying them to fit the notation used in this work, Equations 3.9 and 3.10 are derived:

$$AIC = |S| \cdot \log\left(\frac{\sum_s D_s^2}{|S|}\right) + 2 \cdot (|M| + 1) \cdot |R| \tag{3.9}$$

$$BIC = |S| \cdot \log\left(\frac{\sum_s D_s^2}{|S|}\right) + (|M| + 1) \cdot |R| \cdot \ln(|S|) \tag{3.10}$$

where:

$|M|$ cardinality of set m, representing the total number of input variables

$|S|$ cardinality of set s, representing the total number of samples

$|R|$ cardinality of set r, representing the total number of regions

Both equations can be analysed through two terms, model accuracy and complexity. Model accuracy is quantified using the residual sum of squares, $\sum_s D_s^2$, which is the deviation between prediction and truth. In both criteria the complexity term is $(|M| + 1) \cdot |R|$ since each regression model has $|M|$ coefficients plus one intercept and it is multiplied by the total number of regions $|R|$. The difference of the two criteria is the stricter penalty that the *BIC* imposes to the complexity term. It uses a multiplication factor which depends on the size of the dataset.

For each added region, the value of the criteria changes since the overall model is expected to have better accuracy and as a result a smaller $\sum_s D_s^2$ value but also a larger penalty. So, by minimising the criteria, a balance between complexity and accuracy is achieved.

This iterative approach is using the mathematical model described in Section 3.2.1 and post-processes the results using either the *AIC* or the *BIC* as a termination criterion. Depending on which metric is used, the variant is called *Piecewise Regression with Iterative Akaike information criterion* (`PRIA`) or *Piecewise Regression with Iterative Bayesian information criterion* (`PRIB`), and can be summarised as follows:

- minimise objective function 3.8

subject to:

- constraints 3.1-3.7
- post-process the results with 3.9 or 3.10.

The optimisation model still requires a specific partitioning variable as input. An iterative approach is used to select that partitioning variable as follows:

The number of regions is fixed to $R = 2$. Then the optimisation model of Section 3.2.1 is solved separately for each input variable. The selected variable is the one that yields the minimum error. It is worth noting that it is not necessary to use the information criteria for this step, since the complexity for all the different models is the same (number of regions fixed to $R = 2$). So, the only factor affecting the results is the fitting error.

The next step of the algorithm is to identify the optimal number of regions. The algorithm again solves multiple *MILP* models, but this time the partitioning variable is fixed and with each iteration a new region is added. At the end of each iteration, the values of the criteria are checked and compared with the ones from the previous iteration. If there is an improvement (*AIC* or *BIC* value decreases) then the algorithm iterates again. Otherwise, the iterations are terminated.

This heuristic approach that is described in Figure 3.2 for terminating the algorithm is very similar to the one used in the original `OPLRA` work. The major difference is the use of the *AIC* and *BIC* in the final loop. This loop replaces the original stopping criterion with the aim of overcoming over or under fitting.

Yang et al. (2016) introduced a user-specified parameter, called β, as a threshold to stop the iterations and converge to a solution. Assigning a value to that parameter required a sensitivity analysis that was performed using specific examples. At the end of the analysis, a single value was chosen for this parameter and it would be used for all future input data. This action has an effect on overall performance since the algorithm was tailored around those specific datasets and addressing this issue is a key improvement of the new proposed methods.

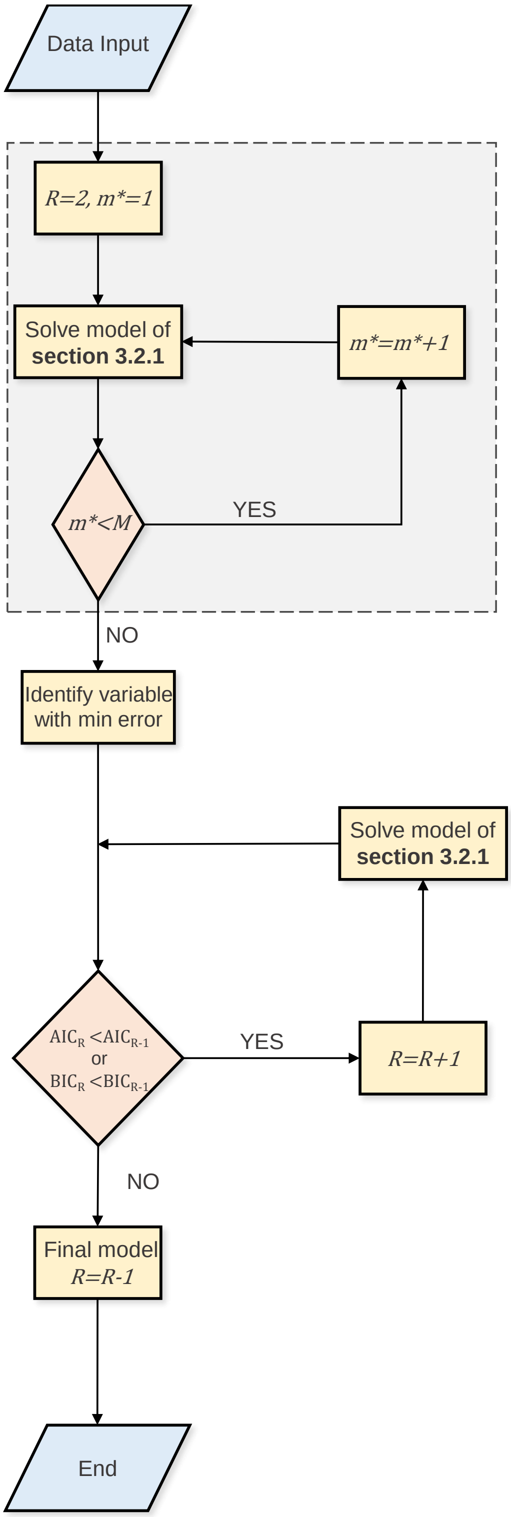

Data Input
R=2, m*=1
Solve model of section 3.2.1
m*=m*+1
m*<M
YES
NO
Identify variable with min error
Solve model of section 3.2.1
AIC_R <AIC_R-1 or BIC_R <BIC_R-1
YES
R=R+1
NO
Final model R=R-1
End

FIGURE 3.2: Flowchart for the proposed iterative approaches

3.2.3 Single-level MILP approaches

In this section, a single-level *MILP* mathematical model is formulated which extends the optimisation model of Section 3.2.1. In this context, single-level means that the mathematical model can now optimise, in a single MILP, both the number of regions as well as the break points and the regression coefficients.

To achieve this, a new set of binary variables is introduced, E_r, to represent the number of selected or "active" regions. Constraints are introduced to formulate the "activation" of regions and the assignment of samples to them. However, as stated the goal is to formulate the optimisation model as an *MILP* and the non-linear expression of the criteria poses an obstacle.

The answer to overcoming this obstacle is to reformulate the criteria and approximate their values by using linear expressions. The selected approach is to use a set of piecewise linear expressions to approximate the logarithm function. In order to further simplify the formulation, instead of using the residual sum of squares, the formulation will use the sum of the absolute deviation. The linear approximation and the added model constraints are explained below:

Indices

i	break points for the piecewise expressions of the approximation

Parameters

γ_i	discrete points for the linearisation
β_i	"output" of the discrete points

Variables

AIC	*Akaike Information Criterion* value
BIC	*Bayesian Information Criterion* value
G	approximated value of the logarithm

Binary variables

E_r	1 if region r is selected; 0 otherwise

SOS2 Variables

λ_i	discrete points for the linear approximation

Mathematical Constraints

Given a maximum number of allowable regions R, the optimisation model will select to "activate" the number of regions that are needed in order to minimise the value of the information criteria. The following constraint, ensures that if a region r is active, $E_r = 1$, then the next region can be either active or not. However, if a region r is not active, $E_r = 0$, then the next $r + 1$ region is forced to be deactivated, $E_{r+1} = 0$. As a result, all the subsequent regions will also be deactivated.

$$E_{r+1} \leq E_r \qquad \forall\, r = 1, 2, ..., R-1 \tag{3.11}$$

Once a region has been activated, samples can be allocated to this region. The following constraint ensures that if a region r is active, $E_r = 1$, then the F_{sr} binary variables for all the samples s are free to be either 0 or 1. However, if a region is not active, $E_r = 0$, all the F_{sr} variables for that region are forced to 0, ensuring that no samples will be allocated to this "inactive" region.

$$F_{sr} \leq E_r \qquad \forall\, r, s \tag{3.12}$$

The next set of equations is responsible for the approximation of the logarithm function in the *AIC* and *BIC*. The approximation is achieved by piecewise linear expressions. A set of *SOS2* variables (special ordered set of type 2, which means that at most two variables within this ordered set can take on non-zero values) is introduced in order to select the correct linear expression. Parameters β and

γ are used to discretise the domain of the function in a given range.

$$\beta_i = \ln(\gamma_i) \qquad \forall i \tag{3.13}$$

$$\sum_s D_s = \sum_i \gamma_i \cdot \lambda_i \tag{3.14}$$

$$G = \sum_i \beta_i \cdot \lambda_i \tag{3.15}$$

$$\sum_i \lambda_i = 1 \tag{3.16}$$

Figure 3.3 is an example to illustrate how the approximation works. The "true" natural logarithm function is represented by the red line. The first step is to choose the number of points for discretising the domain. The higher the number of points, the higher the accuracy of the approximation at the cost of computational expense. In this illustration, five points are selected (set i has 5 elements) with the values of γ =10, 20, 40, 60 and 80 respectively. The next step is to calculate the corresponding logarithm values, which are the values of the β parameter. As a visual aid, the piecewise linear expressions have also been plotted.

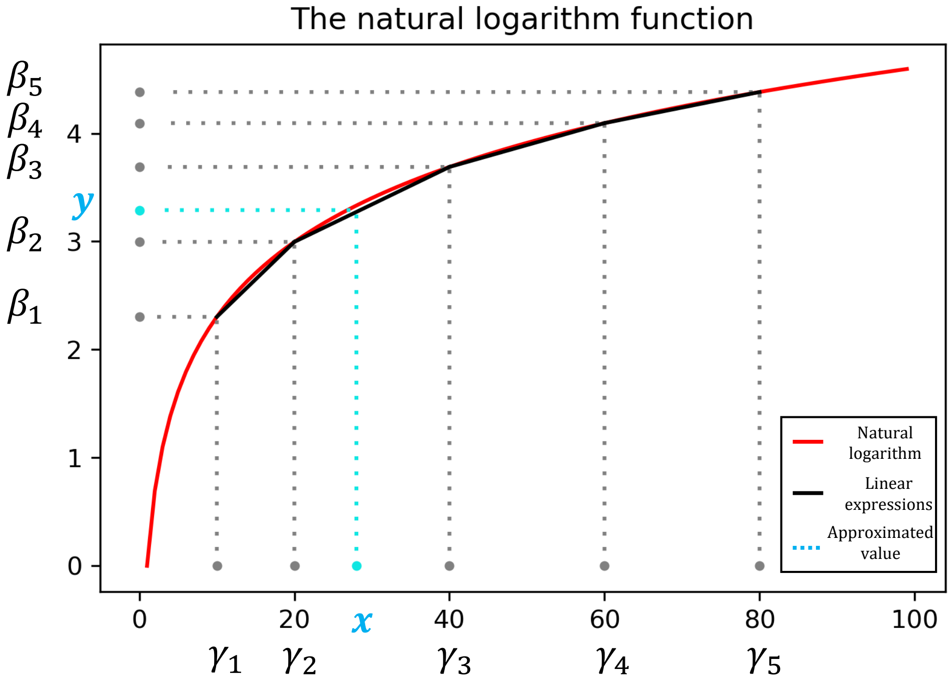


FIGURE 3.3: Illustration of the linear approximation of the logarithm function

Suppose that we want to approximate the point $x = 28$. Point x is between γ_2 and γ_3, so as a result variables λ_2 and λ_3 will have a non-zero value. The values of those variables will determine the approximated value y as follows:

$$\begin{aligned} &x = \gamma_2 \cdot \lambda_2 + \gamma_3 \cdot \lambda_3 \\ &y = \beta_2 \cdot \lambda_2 + \beta_3 \cdot \lambda_3 \\ &\lambda_2 + \lambda_3 = 1 \end{aligned}$$

substituting the numbers:

$$\begin{aligned} &28 = 20 \cdot \lambda_2 + 40 \cdot \lambda_3 \\ &y = 3 \cdot \lambda_2 + 3.7 \cdot \lambda_3 \\ &\lambda_2 + \lambda_3 = 1 \end{aligned}$$

This is a system of 3 equations with 3 unknown variables. Solving this system of equations leads to:

$$\lambda_2 = 0.6,\ \lambda_3 = 0.4,\ y = 3.3$$

A simple way to interpret the result is that the final approximated value is 60% the value of β_2 and 40% the value of β_3.

In the formulation of Equations 3.11-3.16, the value x that needs to be approximated is the $\sum_s D_s$, whereas the logarithm approximation y is represented with variable G.

Finally, formulating the objective function depends on the criterion that is chosen in order to perform model selection. For the *AIC* case:

$$\min AIC = |S| \cdot G - |S| \ln(|S|) + 2(|M| + 1) \cdot \sum_r E_r \tag{3.17}$$

Whereas for the *BIC* case:

$$\min BIC = |S| \cdot G - |S| \ln(|S|) + \ln |S| \cdot (|M| + 1) \cdot \sum_r E_r \tag{3.18}$$

The first term of the right hand side of Equations 3.17 and 3.18 is the accuracy of the fitting, whereas the second term is the complexity of the model. Equations 3.17 and 3.18 have different penalty terms, with *BIC* being more strict since it depends on the size of the examined dataset. The term $|M| + 1$ is also part of model complexity, since there are $|M|$ regression coefficients plus 1 intercept. Finally, depending on the selected number of active regions, more coeffficients are added to the model. Hence, the term $\sum_r E_r$ is essential to account for the number of regions.

In this approach, the partitioning variable is selected the same way as in Section 3.2.2. The next step is to specify the maximum number of regions R and the set of binary variables E_r will decide the optimal number of regions. Overall, the proposed *MILP* model can be split into two sub-models depending on the objective function. The first model is the *Piecewise Regression with Optimised Akaike Information Criterion* (PROA) and the *Piecewise Regression with Optimised Bayesion Information Criterion* (PROB) and can be summarised as follows:

- minimise objective function 3.17 or 3.18
- subject to constraints 3.1- 3.7 and 3.12- 3.16

Table 3.1 is a brief summary of the optimisation based regression approaches.

TABLE 3.1: A summary of the of the otpimisation based approaches

Method	**Description**	**Equations**
OPLRA	Original OPLRA model (Yang et al., 2016)	min Eq. 3.8, s.t Eq. 3.1- Eq. 3.7
PRIA	OPLRA model with *AIC* post-process	min Eq. 3.8, s.t Eq. 3.1- Eq. 3.7, post Eq. 3.9
PRIB	OPLRA model with *BIC* post-process	min Eq. 3.8, s.t Eq. 3.1- Eq. 3.7, post Eq. 3.10
PROA	Single level MILP with *AIC* objective function	min Eq. 3.17, s.t. Eq. 3.1-Eq. 3.7 and Eq. 3.12 - Eq. 3.16
PROB	Single level MILP with *BIC* objective function	min Eq. 3.18, s.t. Eq. 3.1-Eq. 3.7 and Eq. 3.12 - Eq. 3.16

3.3 Numerical and computational section

3.3.1 Algorithm implementation

The R v3.3.1 programming language was chosen for the implementation of this algorithm. The language is free and open source with good support from the community and it includes many powerful and popular libraries that are related to machine learning. However, for the optimisation parts of the algorithm, GAMS v24.7.1 and the solver CPLEX v.12.6.3.0 were chosen. The following figure illustrates the implementation of the algorithm:

Figure 3.4 illustrates the implementation and the integration of GAMS and R. All the blue elements and boxes have been implemented in R while the yellow ones in GAMS.

The user has to specify the desired approach (the options are listed in table 3.1) and provide the input dataset. The next step of the implementation is to pre-process the data and transform them into the correct format for conversion. It is worth noting that this pre-processing step **does not** perform any data-cleaning or feature engineering/selection methodologies. It simply handles the data in order to prepare them for the next step.

The pre-processing step captures all the necessary information that is required for the optimisation models that are described in Section 3.2. The next step is to convert all of the data into an appropriate GAMS format called gdx, using a package called gdxrrw (Jain and Dirkse, 2018).

The implementation then calls two GAMS scripts which are responsible for identifying the partitioning variable, as described in Figure 3.2, and then find the optimal number of regions depending on which method was selected. Once this is done, the results are imported back into R, where they are transformed to a user-friendly format. A custom function for making predictions on new samples has also been implemented entirely in R.

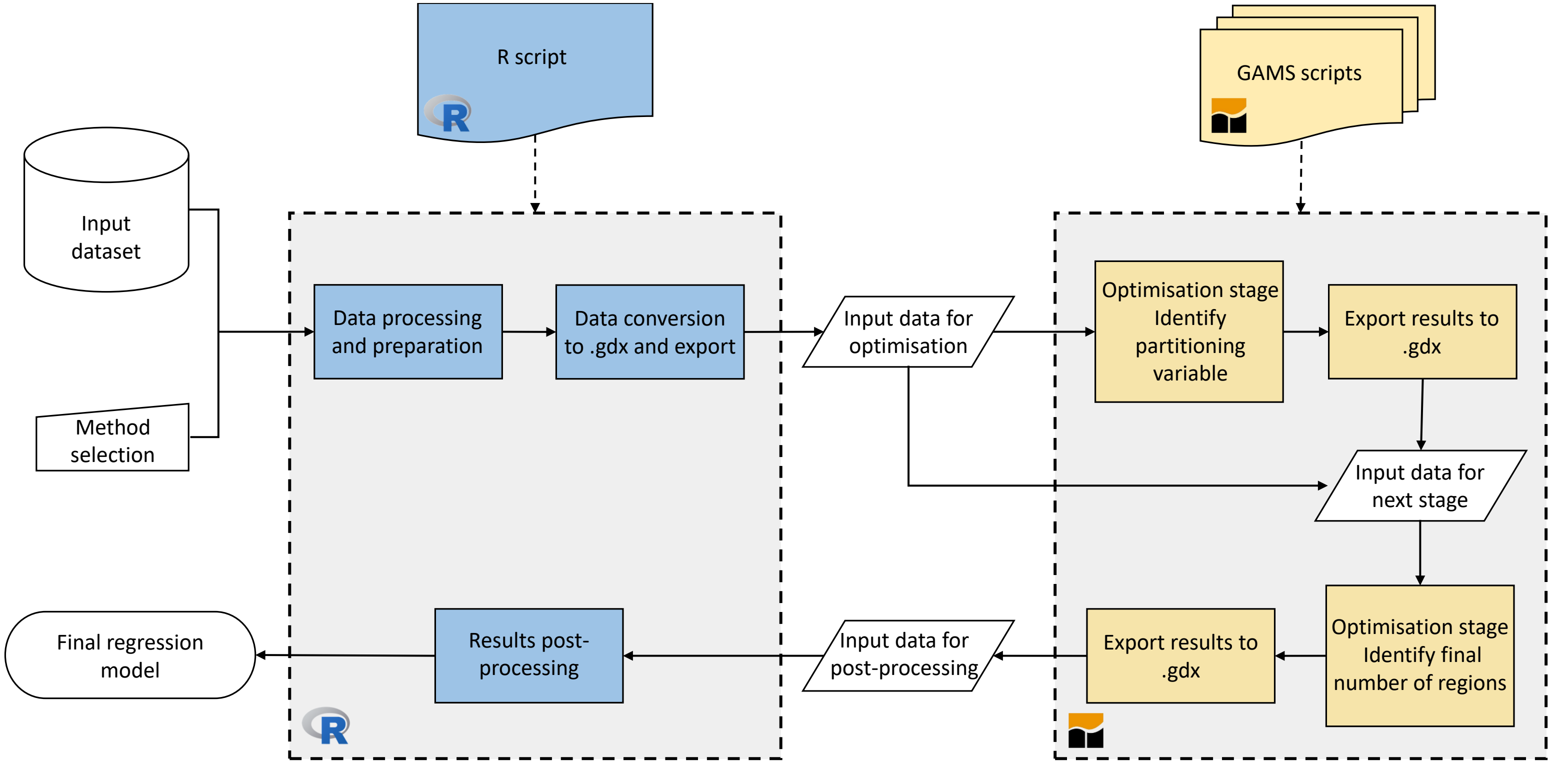

R script
GAMS scripts
Input dataset
Method selection
Data processing and preparation
Data conversion to .gdx and export
Input data for optimisation
Optimisation stage Identify partitioning variable
Export results to .gdx
Input data for next stage
Optimisation stage Identify final number of regions
Export results to .gdx
Input data for post-processing
Results post-processing
Final regression model

FIGURE 3.4: The implementation of the piecewise regression algorithm

3.3.2 Illustrative example

An example from literature is used to demonstrate the final form of the regression equations. This example is about the octane rating of fuel. This specific dataset investigates the octane rating of petrol during a manufacturing process in a refinery. The rating of the fuel is measured as a function of 3 raw materials, denoted A1, A2 and A3, and a variable that quantifies the manufacturing conditions of the refinery, denoted Q (Wood, 1973).

TABLE 3.2: Final regression functions for some of the proposed methods

Method	Regression functions
PRIA	$Y = \begin{cases} -5.13 \cdot A_1 + 0.11 \cdot A_2 - 0.71 \cdot A_3 + 1.95 \cdot Q + 95.71, & 0 \leq A_3 \leq 0.58 \\ -7.57 \cdot A_1 - 0.97 \cdot A_2 - 3.70 \cdot A_3 + 3.96 \cdot Q + 98.83, & 0.58 < A_3 \leq 0.71 \\ -8.13 \cdot A_1 - 1.80 \cdot A_2 - 1.82 \cdot A_3 + 2.05 \cdot Q + 99.00, & 0.71 < A_3 \leq 0.92 \\ 23 \cdot A_1 + 2.95 \cdot A_2 + 13.90 \cdot A_3 + 6.48 \cdot Q + 59.15, & 0.71 < A_3 \leq 1 \end{cases}$
PROA	$Y = \begin{cases} -5.13 \cdot A_1 + 0.11 \cdot A_2 - 0.71 \cdot A_3 + 1.95 \cdot Q + 95.71, & 0 \leq A_3 \leq 0.58 \\ -7.79 \cdot A_1 - 1.19 \cdot A_2 - 0.1.07 \cdot A_3 + 3.11 \cdot Q + 97.71, & 0.58 < A_3 \leq 1 \end{cases}$

Table 3.2 is an illustration of the different regression models that are produced by the iterative and single-level approaches using *AIC*.

Both approaches were able to identify exactly the same linear expression for the first region. This linear expression is a function of all four variables with the corresponding regression coefficients and the intercept. However, the iterative approach identifies a total of four regions compared to just two of the single level *MILP* approach.

Even though both models use exactly the same criterion, the results are not the same. In order to determine which of the proposed methods has the best predictive performance, more testing is required. In the next section a number of examples are used to test and compare the proposed methods to other literature algorithms.

3.3.3 Datasets examined in this work

To test the proposed methods a number of real world datasets have been used. The datasets reported in table 3.3 are derived from numerous online sources. Specifically, the pharmacokinetics and earthquake data are available through the `datasets` package in `R`, bodyfat and sensory data are available through *StatLib* (Vlachos, 2005), distillation data from *OpenMV.net* and the rest from the *UCI* machine learning repository (Dheeru and Karra-Taniskidou, 2017)

TABLE 3.3: Regression datasets examined in piecewise regression work

Dataset	**No.samples**	**No.variables**
Pharmacokinetics	132	4
Bodyfat	252	14
Distillation	253	26
Yacht Hydrodynamics	308	6
Sensory	576	11
Cooling efficiency	768	8
Heating efficiency	768	8
Earthquake	1000	4
Concrete	1030	8
White wine quality	4898	11

The **yacht** hydrodynamics set predicts the residuary resistance of sailing yachts to evaluate the ships' performance and estimate the required propulsive power. The energy efficiency dataset (Tsanas and Xifara, 2012) assesses the **heating** and **cooling** load requirements of different buildings as a function of 8 parameters. The **concrete** dataset (Yeh, 1998) tries to predict the compressive strength of concrete as a structural material. The **wine** dataset (Cortez et al., 2009) predicts the quality of white wine based on its properties.

The **pharmacokinetics** dataset contains data from a study of the kinetics of the anti-asthmatic drug theophylline. Twelve subjects were given oral doses of the drug and the aim is to predict the final concentration of theophylline of each subject. The **earthquake** dataset gives the location of seismic events that occurred near Fiji since 1964. The **bodyfat** dataset uses features such as age, weight and height to measure the percentage of bodyfat in a subject. The **sensory** dataset contains data for the evaluation of wine quality by a total of 6 judges. The **distillation** dataset is comprised of measurements from a distillation column, with vapour pressure being the quality variable that was measured in the lab.

3.3.4 Algorithm validation and comparison

A number of regression algorithms from literature are also implemented for comparison purposes. The algorithms include `KNN`, *Random Forest*, `MARS` and `SVM` regression. All of those methods are implemented in the `R` programming language using the `FNN` (Beygelzimer et al., 2013), `randomForest` (Liaw and Wiener, 2002), `earth` (Milborrow, 2018) and `e1071` (Meyer et al., 2017) packages respectively.

In this work, 5-fold cross-validation is selected to evaluate the performance of all the examined algorithms. 10 runs will be performed and the *Mean Absolute Error* (*MAE*) between model prediction and the true data will be calculated for each fold. The final score is the average of all the runs.

It is standard practice in machine learning tasks to perform feature scaling (Muller and Guido, 2016). For each dataset, min-max feature scaling will be applied and the selected range is [0,1] according to Equation 2.1. This scaling will help to eliminate some variable being more dominant than others in the final regression model.

Additionally, scaling the features to a specific range will make the selection of the large positive number U_1 (from the model of Section 3.2.1) easier, since all the features are in the same range.

The iterative methods do not require any input or hyper-parameter tuning from the user. The single-level *MILP* approaches however require an upper bound for the maximum allowable number of regions. For the computational runs of this work, this limit has been set to 5 regions, which is believed to be a suitable upper limit for achieving good performance but at the same time keeping the complexity of the model to reasonable limits.

Furthermore, the hyper-parameters of the established algorithms from the literature are set to their default values. These default values are the ones suggested either by the creators of the algorithms or the developers of the respective packages in `R`.

3.4 Computational results

3.4.1 Cross-validation runs

Table 3.4 contains the *MAE* results of all the 5-fold cross-validation runs. For comparison purposes, the first step is to examine how the new methods perform against the previous work and seek possible improvements. Next, the new methods will be compared to established methods from literature. For each examined dataset, the regression analysis that had the lowest *MAE* score is presented with bold.

`PROA` has the lowest average error on 7 out of the 10 examined datasets compared to `OPLRA`. `PRIA` is also competitive and achieved a better score for multiple datasets. However, it is worth noting that the *BIC*-based algorithms all have worse performance than the *AIC*-based ones.

It is very important to test and compare the accuracy of the proposed methods with established ones from literature. Since `PROA` was the best performer amongst the proposed methods, it is selected to be compared to the established methods. We can see that overall, the proposed `PROA` method has the lowest error in only 5 datasets. However, upon examining the results closer it becomes apparent that the method performs well since the error scores are always very close to the ones that have the best overall performance.

To demonstrate that, a graph is developed comparing the overall performance of the algorithms. This graph assigns a score to each one of the examined algorithms in the scale of [1,10]. The score is calculated by looking at the relative performance of all the algorithms for each dataset. Specifically, for a single dataset, the algorithm that had the best *MAE* score is assigned 10 points,

TABLE 3.4: Cross-validation results using *MAE* for the piecewise regression work

	Yacht	**Cooling**	**Heating**	**Concrete**	**Wine**
OPLRA	0.689	**1.275**	**0.805**	4.845	0.551
PRIA	0.680	1.337	0.820	4.840	0.553
PRIB	0.699	1.342	0.909	4.922	0.567
PROA	**0.678**	**1.275**	0.806	4.838	0.555
PROB	0.688	1.351	0.906	4.920	0.566
KNN	5.788	2.237	2.063	8.924	0.577
SVM	3.673	1.820	1.456	4.864	0.518
RandFor	2.454	1.326	0.861	**4.029**	**0.439**
MARS	1.079	1.340	0.826	4.932	0.569
	Bodyfat	**Sensory**	**Distil**	**Pharma**	**Earthquake**
OPLRA	1.273	0.632	2.650	1.613	**7.238**
PRIA	0.785	0.633	1.110	1.352	7.426
PRIB	0.763	0.652	1.127	1.387	7.357
PROA	0.631	0.626	**1.025**	**1.288**	**7.238**
PROB	1.341	0.636	1.105	1.325	7.256
KNN	2.869	0.642	1.966	1.981	8.464
SVM	1.391	0.613	1.128	1.834	7.250
RandFor	1.532	**0.562**	1.153	1.677	7.978
MARS	**0.389**	0.616	1.147	1.420	7.389

whereas the algorithm that had the worst *MAE* score is assigned 1 point. The rest of algorithms are within this range.

For example, there are 9 examined algorithms in this chapter. Examining the **Yacht** dataset, the *MAE* results of table 3.4 show that PROA has the best score. So, it is awarded 10 points. The second best algorithm for this dataset is PRIA, so it is awarded 8.88 points. The third best algorithm is awarded 7.75 points and so on, until the worst performing algorithm, which in this case is KNN, is awarded 1 point. The final score that is plotted on the graph, is the average score of each algorithm for all the datasets.

The ranking of Figure 3.5 aids with the interpretation and understanding of the relative performance of the regression algorithms used in this work. The PROA algorithm is at the top of the ranking with the highest average score. This demonstrates that the algorithm is able to compete with other established regression algorithms, at least for datasets of similar dimensionality and complexity as the ones examined in this work and in some cases even outperform them.

OPLRA and PRIA have very similar average scores, while the algorithms that use the *BIC* have clearly the worst performance among the proposed algorithms.

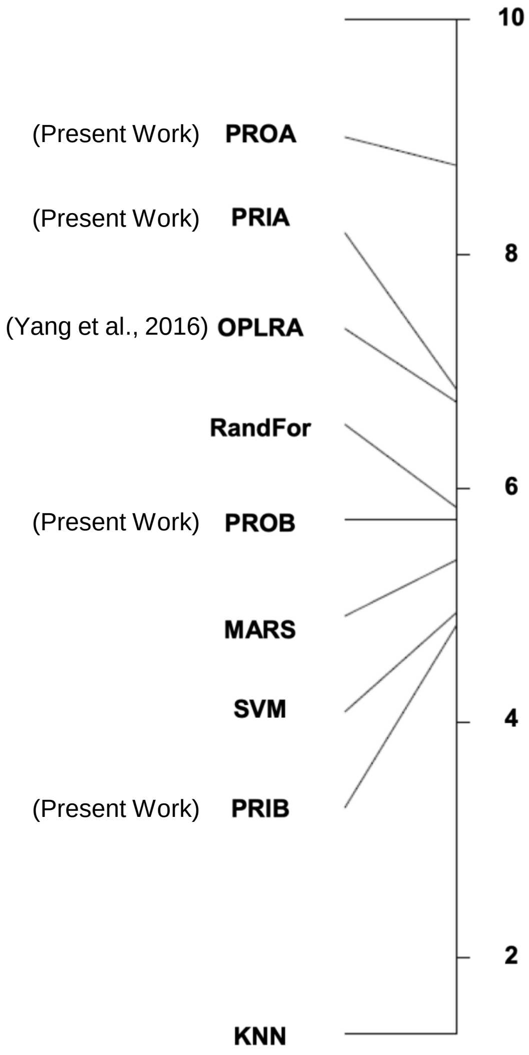


FIGURE 3.5: Visualisation of the performance of the methods based on *MAE* values.

The bad performance of the *BIC*-based algorithms compared to the *AIC* ones could be explained by the different penalty factors the two criteria use. Equation 2.4, which describes *AIC*, uses a multiplication factor of 2 as a penalty for complexity. This factor is fixed and independent of the dataset that is used.

In contrast, Equation 2.5 uses a factor which depends on the size of the dataset. The bigger the dataset, the larger the penalty that is imposed to complexity. Since the *BIC* penalises complexity that much, the regression models have fewer number of regions than what *AIC* produces. As a result, the criterion is not able to perform well.

3.4.2 Statistical analysis

This ranking, despite being helpful and insightful, should not be the only way of evaluating the overall performance. Performing a statistical test is vital in order to check whether the difference between the *CV* results of table 3.4 is statistically significant.

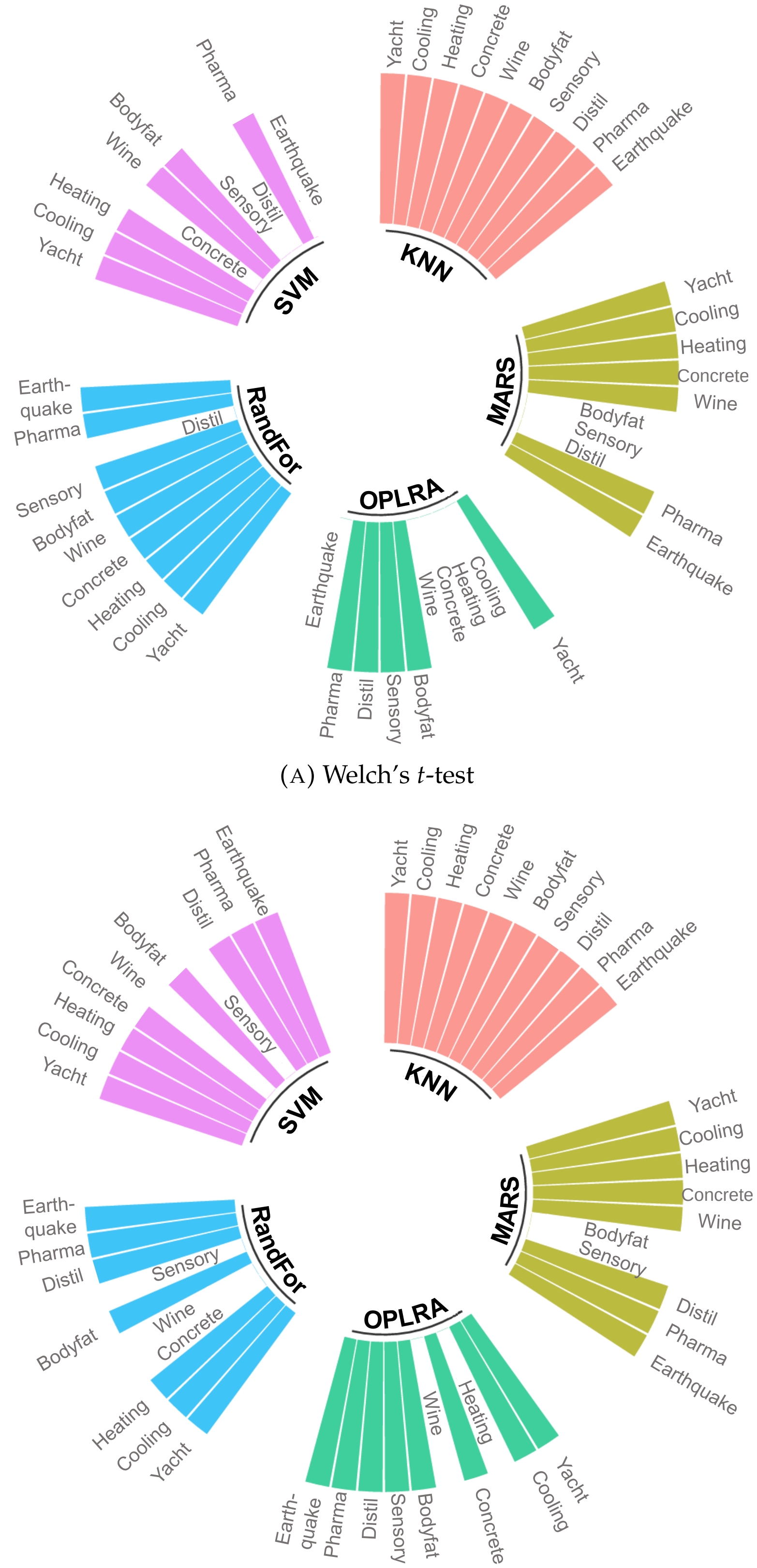
The statistical test chosen for this analysis is the Welch's *t*-test. This is a two-sample test which is used to test the hypothesis that two populations have equal means and is reliable when the samples have unequal variances (Welch, 1947). If there is evidence to reject this hypothesis, then it can be concluded that the difference between the two means is significant.

For each dataset, the two different populations that will be compared are the values of the 10 cross validation runs between the `PROA` 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 between 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.

- Calculate the *t* statistic using eq. B.1
- 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 3.6 is a visual representation of the statistical analysis performed for the *CV* results. The circles contain five groups, one for each competing regression algorithm. Each group has 10 bars that correspond to the 10 examined datasets.



(A) Welch's *t*-test

(B) *MAE* scores

FIGURE 3.6: Visualisation of the computational results of this work. Case (a) is a representation of the statistical analysis using Welch's *t*-test. Case (b) is a representation of the *MAE* performance.

Figure 3.6a 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 *KNN*. There is a significant difference in 9 examples with *Random Forest* and 7 examples both SVM and MARS. Finally, there is a difference in only 5 examples with OPLRA.

Figure 3.6b is similar to Figure 3.6a, but this time a bar is present only if *PROA* has achieved a lower *MAE* score for this specific example. It is clear that the proposed approach has consistently outperformed *KNN* and *OPLRA*. Even though *OPLRA* shares part of the optimisation model, the addition of the *AIC* has greatly improved the results, with *PROA* providing better MAE score in 8 out of the 10 examples. PROA has achieved lower error values for 8 out of the 10 examples compared to SVM and MARS, but against *Random Forest* it performed better for only 7 examples.

To accurately compare those algorithms, both figures should be taken into account. For a specific dataset, it is desirable to have a bar present in both figures since that would suggest strong evidence that the *MAE* averages of all the *CV* runs are indeed statistically different and PROA has better performance. Based on that rule, the proposed algorithm provided better results than OPLRA, MARS and KNN, while outperforming for the most part *Random Forest* and SVM.

Figure 3.7 below is a visual representation of the performance of PROA compared to the other methods. Each sub-figure is made of two graphs. The outer ring represents the percentage of examples for which there was a statistically meaningful difference in the *CV* results between PROA and another algorithm. The pie chart in the middle represents the percentage of datasets for which PROA outperforms the competing regression algorithm, provided that there is a statistically meaningful difference in the results.

It is clear that when comparing PROA to KNN, there is a statistical difference for all 10 datasets and PROA is the winner in all 10. PROA also performed very well compared to MARS since there was a statistical difference in the majority of the examined datasets and PROA had lower error values in all of them.

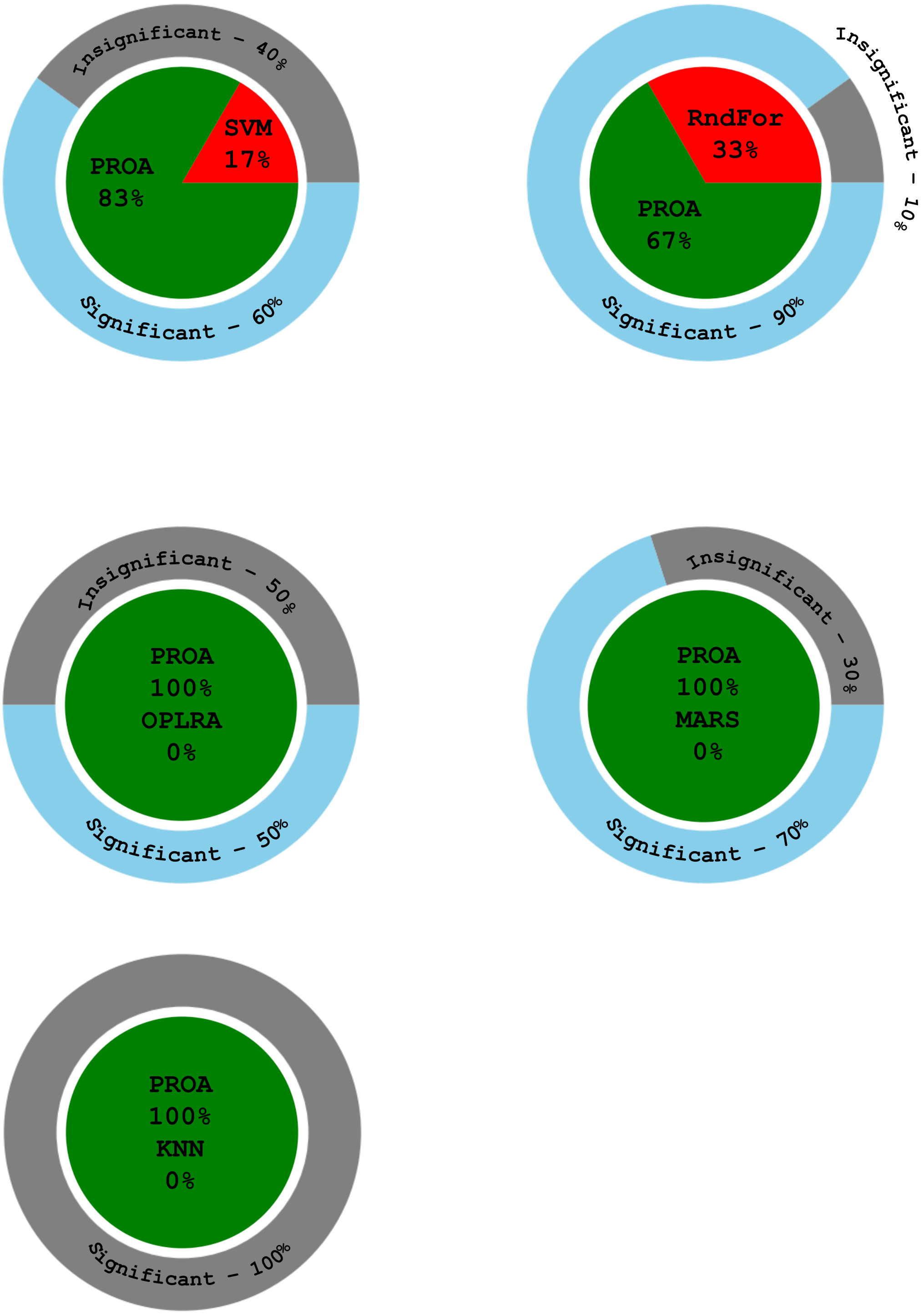


FIGURE 3.7: Percentage of winning between PROA and the other methods, based only on the datasets with meaningful statistical difference.

Comparing the proposed algorithm with OPLRA is interesting. There is a statistical difference in only half of the examined datasets, but PROA is clearly outperforming OPLRA on the other half. This can be attributed to two facts. The first fact is that those two algorithms share part of the optimisation model that they use to fit the data. The second fact is that some of the examined examples were also used in the original work. This is a very important detail which indicates that when the OPLRA algorithm deals with new datasets, it struggles to generalise well. It was mentioned at Section 1.3.1 that this behaviour can be attributed to the heuristic approach that was used by the original authors, which leads to potential overfitting models.

PROA is able to compete against SVM by providing lower error values for the majority of the statistically significant datasets. However, there is a significant difference only for 6 out of the 10 examined examples. Finally, *Random Forest* is the biggest competitor for the examined examples, achieving lower error values for 33% of the statistically significant datasets.

3.5 Concluding remarks

The work in this chapter proposes a piecewise regression algorithm. This algorithm uses the optimisation model of the existing OPLRA algorithm and extends its formulation to include information criteria. The information criteria are taken into consideration by two different ways. The first one is an iterative post-process approach, where the values of the criteria are calculated after the optimisation stage. The second way directly optimises the values of the criteria, since they are the objective functions of the corresponding optimisation models.

The inclusion of the criteria is the key contribution of this novel algorithm. The task of identifying the optimal number of required regions is now decided by the minimisation of the information criteria. The first step of the proposed variants is the selection of the best partitioning features. For the iterative approaches, the next step is the execution of the optimisation model of Section 3.2.1, by adding a new region at each iteration. The iterations stop when the post-processed values of the criteria stop decreasing. For the single-level *MILP* approaches, given a maximum number of allowable regions, the optimal number is decided by the execution of a single model.

The predictive performance of the algorithm is tested through 10 runs of 5-fold cross-validation. The datasets used in this work are real-world examples that are available through online public repositories. Various well-established algorithms are used for comparison purposes. The results indicate that the new algorithm, especially the `PROA` variant, is a good regression alternative to other established algorithms. The validity of the results is proven by performing a statistical analysis to check for significance. Furthermore, the algorithm has the added benefit of using linear models for each region which are easy to understand, and the number of regions decided by the model require minimal user input.

The core concept of the optimisation model is used in the next chapter to develop a decision tree regressor. Instead of splitting the data into multiple regions, the regressor creates binary splitting rules. The use of information criteria and their formulation as an *MILP* model are utilised again and applied to solve the problem of identifying the “optimal” partitioning feature.

4 Development of a decision tree regressor

The work in this chapter deals with the development of a decision tree regressor. A description of an existing algorithm is given and its optimisation model is used in order to create the novel method. An established statistical test is introduced and a novel variable selection optimisation model is developed to create the new decision tree algorithm. The proposed approach is tested using data that were retrieved from online sources and its performance is compared to various established tree algorithms from the literature.

4.1 Introduction

Decision trees can be used to visually and explicitly represent decision making by using tree-like models. Decision trees are also widely used in machine learning for both regression and classification tasks (Muller and Guido, 2016).

In machine learning, tree models employ a recursive binary splitting approach of the input data in order to generate a tree-like structure. Depending on the task at hand, the splitting approach minimises a cost function which tries to find the most homogeneous branches. Stopping the tree generation process can involve rules such as setting a minimum number of training samples on each terminal node or setting a maximum depth size, which refers to the length of the longest path from the root to a leaf. Many algorithms also apply a pruning step that reduces the final size of the constructed tree by removing branches that have low importance and therefore reducing the risk of overfitting.

The work in this chapter describes the development of a tree regression algorithm that uses an *MILP* formulation to optimally split nodes and a well established statistical test for assessing the tree generation process. Finally a novel mathematical formulation of a subset selection model to identify a subset of candidate variables to be considered for binary splitting.

4.2 Proposed algorithm

This section proposes two variations of a novel regression algorithm that is based on the partitioning model that is presented in Section 4.2.1. That model is part of the `MPtree` algorithm and it is responsible for optimally splitting a node into two child 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 introduces a new mathematical formulation that minimises the value of the *BIC* to select an optimal subset of input variables to be considered for splitting.

4.2.1 Mathematical formulation of MPtree (Yang et al., 2017)

In this section, the mathematical programming model that was used in the `MPtree` algorithm is described as formulated by Yang et al. (2017). This model accepts a multivariate dataset as input and splits it into two nodes on a specific variable and fits a regression model with quadratic terms to each new node. The model calculates the optimal regression coefficients and intercepts of the regression while minimising the summation of the absolute deviation of the fitting. The mathematical model is presented below:

Indices

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

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

Variables

B_c	intercept of regression function in child node c
Pr_{sc}	predicted output for sample s in child node c
$W1_{mc}$	regression coefficient for feature m in child node c; for linear terms
$W2_{mc}$	regression coefficient for feature m in child node c; for quadratic terms
X_m	break-point value on feature m

Positive variables

D_s	training error between predicted output and observed output for sample s

Binary variables

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

Mathematical Constraints

This model is based on the OPLRA model described in Section 3.2.1. The difference this time is that instead of having multiple regions, the model now is splitting into two nodes.

The two following constraints formulate the assignment of samples to one of the child nodes. Binary variables are introduced for the assignment of samples into the child nodes. Equation 4.1 is responsible for allocating samples to the left child node whereas Equation 4.2 to the right.

$$A_{sm} \leq X_m + U_1 \cdot (1 - F_{sc}) - \epsilon \qquad \forall\, s \in S_n,\ c = left,\ m = m^* \tag{4.1}$$

$$X_m - U_1 \cdot (1 - F_{sc}) + \epsilon \leq A_{sm} \qquad \forall\, s \in S_n,\ c = right,\ m = m^* \tag{4.2}$$

The following constraint formulates that each sample belongs only to one child node. This is achieved by forcing the summation of the binary variables F_{sc} for both child nodes to be equal to 1, for each sample s.

$$\sum_{c \in C_n} F_{sc} = 1 \qquad \forall s \in S_n \tag{4.3}$$

A regression model is fitted to each child node c. Each regression function contains both linear and quadratic terms in order to improve predictive accuracy.

$$Pr_{sc} = \sum_m A_{sm}^2 \cdot W2_{mc} + \sum_m A_{sm} \cdot W1_{mc} + B_c \qquad \forall\, s \in S_n,\ c \in C_n \tag{4.4}$$

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. This can be expressed with the following two constraints:

$$D_s \geq Y_s - Pr_{sc} - U_2 \cdot (1 - F_{sc}) \qquad \forall\, s \in S_n,\ c \in C_n \tag{4.5}$$

$$D_s \geq Pr_{sc} - Y_s - U_2 \cdot (1 - F_{sc}) \qquad \forall\, s \in S_n,\ c \in C_n \tag{4.6}$$

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

$$\min \sum_{s \in S_n} D_s \tag{4.7}$$

The resulting model can be summarised as:
objective function (4.7)
subject to (4.1)-(4.6) constraints
and is formulated as an MILP problem.

4.2.2 Variant I: Introduction of a statistical test

The Chow statistical test

In regression analysis, the F statistical test can be used to assess the quality of segmented regression models. Assuming there is a structural break in the data, splitting them and fitting mathematical functions to the subsets can improve prediction accuracy. 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 (Chow, 1960).

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.

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

$$F = \frac{\left(\dfrac{RSS_1 - RSS_2}{p_2 - p1}\right)}{\dfrac{RSS_2}{n - p_2}} \tag{4.8}$$

where:

RSS_1	residual sum of squares of model 1 (single regression for the entire dataset)
RSS_2	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	total 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 4.8
- 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.

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.

As with other tree algorithms, recursive splitting is used to generate the tree. For each node, the partitioning model of Section 4.2.1 is applied to split nodes into two child nodes. Then the Chow test is applied to compare the linear regression model with the segmented regression model. If there is 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 current node will no longer be considered for splitting and becomes a leaf. 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 Section 4.2.1 to split variables 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 steps until there are no more candidate nodes left to be checked.

4.2.3 Variant II: Variable selection for splitting nodes

According to the proposed algorithm in the previous section, identifying the best partitioning variable employs an exhaustive search approach. Multiple *MILP* models are solved in order to select the partitioning variable that yields the minimum error which 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 subset of variables of the original input space. This leads to potential improvements in prediction accuracy and interpretability. However, one of the main reasons for applying a feature selection method to the proposed tree regression algorithm is to avoid the exhaustive search for the identification of the partitioning variable. In that case, the optimisation model of Section 4.2.1 will go through a set of candidate variables, greatly reducing the overall training time.

Jian et al. (2017) proposed a variable selection method through mixed integer quadratic programming by utilising the *BIC*, whereas Miyashiro and Takano (2015) developed a mixed integer programming approach to deal with the problem of subset selection using *Mallows'* C_p.

The feature selection model that is presented below is an *MILP* formulation. Decision variables are included to determine which input variables should be included in the final subset. The selection is based on fitting a linear regression model to the data while minimising the *BIC* to determine the final number of selected variables. The formulation of the *BIC* is based on the formulation of the model in Section 3.2.3.

Indices

s	data samples, s=1,2,...,S
m	features/input variables $m = 1, 2, .., M$
i	break points for the piecewise expressions of the approximation

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

Variables

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

Positive Variables

D_s	training error between predicted output and observed output for sample s

Binary variables

Z_m	1 if feature m is selected; 0 otherwise

SOS2 variables

λ_i	discrete points used for the linear approximation

Mathematical Constraints

This implementation of feature selection employs binary variables, Z_m, that will decide which features should be selected in the final subset. A linear regression model is fitted to the data based on the following equation.

$$Pr_s = \sum_m A_{sm} \cdot W_m + B \qquad \forall\, s \tag{4.9}$$

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

$$D_s \geq Y_s - Pr_s \qquad \forall\, s \tag{4.10}$$

$$D_s \geq Pr_s - Y_s \qquad \forall\, s \tag{4.11}$$

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

$$W_m \geq LO \cdot Z_m \qquad \forall\, m \tag{4.12}$$

$$W_m \leq UP \cdot Z_m \qquad \forall\, m \tag{4.13}$$

Additional 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, enabling the user to control the size of the selected subset.

$$\sum_m Z_m \geq 1 \tag{4.14}$$

$$\sum_m Z_m \leq N \tag{4.15}$$

Following the formulation of the model in Section 3.2.3, the logarithm function will be approximated through the use of piecewise linear expressions. Set i represents the number of breaking points and parameters β and γ define the actual breaking points for the approximation. Finally, *SOS2* variables are introduced to decide the "correct" linear expression (see Figure 3.3).

$$\beta_i = \ln \gamma_i \qquad \forall\, i \tag{4.16}$$

$$\sum_s D_s = \sum_i \gamma_i \cdot \lambda_i \tag{4.17}$$

$$G = \sum_i \beta_i \cdot \lambda i \tag{4.18}$$

$$\sum_i \lambda_i = 1 \tag{4.19}$$

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

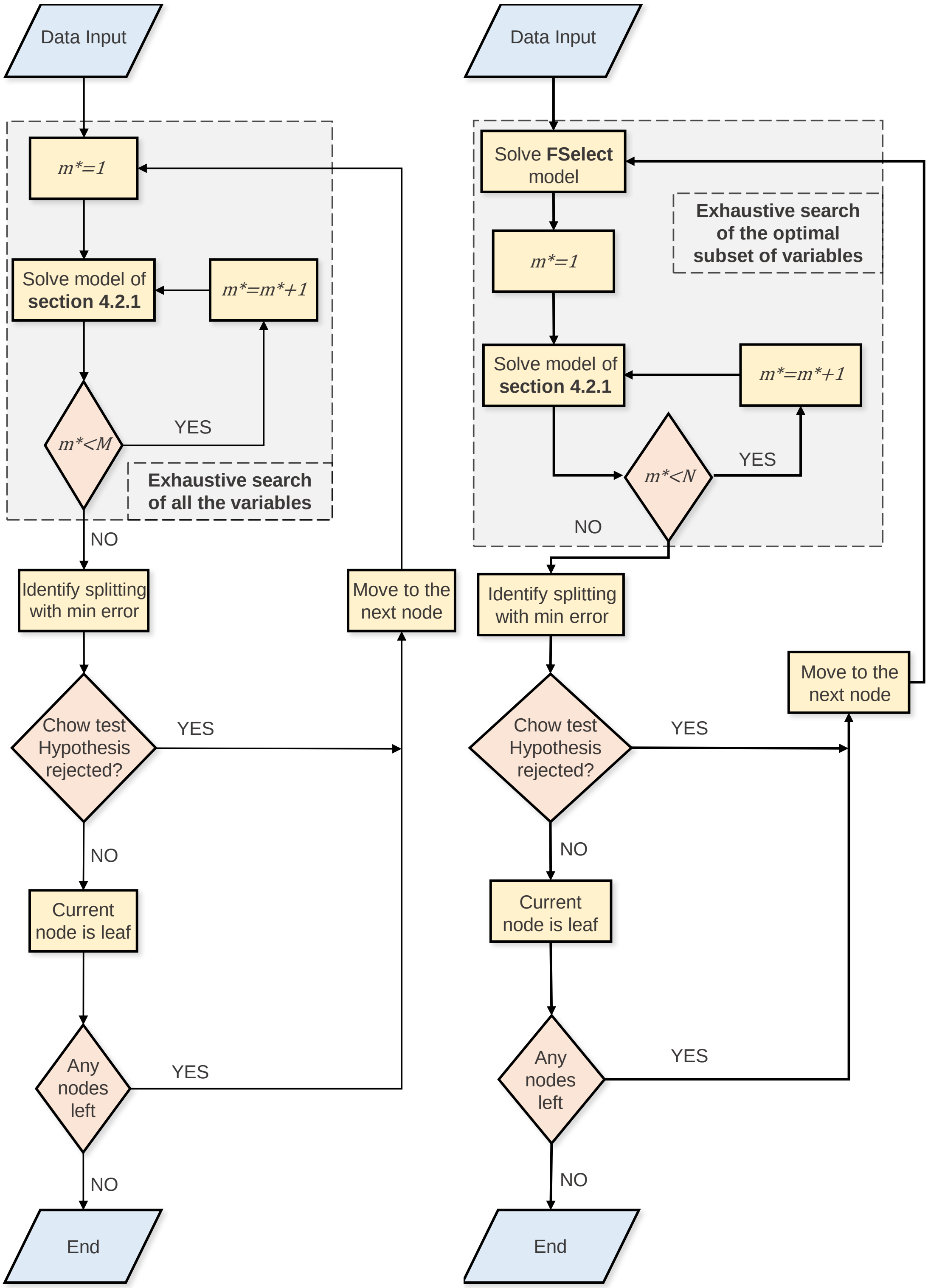
$$\min BIC = |S| \cdot G - |S| \cdot \ln |S| + \ln |S| \cdot \left(\sum_m Z_m + 1 \right) \tag{4.20}$$

$|S|$ is the total number of samples in the dataset and G is the piecewise linear approximation of the logarithm. As mentioned in Section 2.4.1, the minimisation of the BIC results in a balance between model accuracy and model complexity. In this instance, that balance translates to creating a linear regression model that is as accurate as possible but at the same time uses the least number of input variables. The complexity of the model, which is the last term of Equation 4.20, is the total number of selected variables $(\sum_m Z_m)$ plus an extra degree for the intercept of the regression.

The resulting model, from now on known as `FSelect`, is an *MILP* formulation that can be summarised as:

- minimise objective function 4.20
- subject to constraints 4.9-4.19

A maximum number of selected variables is specified by the user with the final number being decided by the optimisation model.


Data Input
m*=1
Solve model of section 4.2.1
m*=m*+1
m*<M
YES
Exhaustive search of all the variables
NO
Identify splitting with min error
Move to the next node
Chow test Hypothesis rejected?
YES
NO
Current node is leaf
Any nodes left
YES
NO
End
Data Input
Solve FSelect model
Exhaustive search of the optimal subset of variables
m*=1
Solve model of section 4.2.1
m*=m*+1
m*<N
YES
NO
Identify splitting with min error
Move to the next node
Chow test Hypothesis rejected?
YES
NO
Current node is leaf
Any nodes left
YES
NO
End

(A) `StatTree` - Exhaustive search

(B) `StatTree` with `FSelect`

FIGURE 4.1: 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.

4.3 Numerical and computational section

4.3.1 Algorithm implementation

Once again, a combination of R and GAMS was chosen for the implementation of the proposed algorithm, illustrated in Figure 4.2. All the blue boxes are implemented in R while the yellow ones in GAMS. This time, the user can specify whether he desires to use the **FSelect** variable selection model. If not, then the StaTree version with exhaustive search is selected by default. The first step of the algorithm is to prepare the data for conversion to gdx format. Once this is done, GAMS is called for with the optimisation part.

If selected by the user, the first optimisation stage is the subset selection *MILP* model. The results are exported to gdx and imported to the next optimisation stage which splits nodes into two subsets. This script contains a loop that checks to find the best partitioning variable based on the subset selected in the previous step (or exhaustive search in case the **FSelect** option is not chosen).

The results are exported to gdx and imported into R, where the *F* test is applied to assess the splitting. If the null hypothesis is rejected, then the splitting is approved and the next available node is selected to start the optimisation process again. At this point, the difference between the implementation of Figure 3.4 and this one becomes apparent. In Figure 3.4, GAMS was responsible for the loop over the set of regions and deciding the optimal number. However, this time GAMS is called through R multiple times, each time checking a single node partitioning without any post-processing of the results. The final step is to post-process the results and transform them into a readable format for the user. A custom function for making predictions on new samples has also been implemented entirely in R.

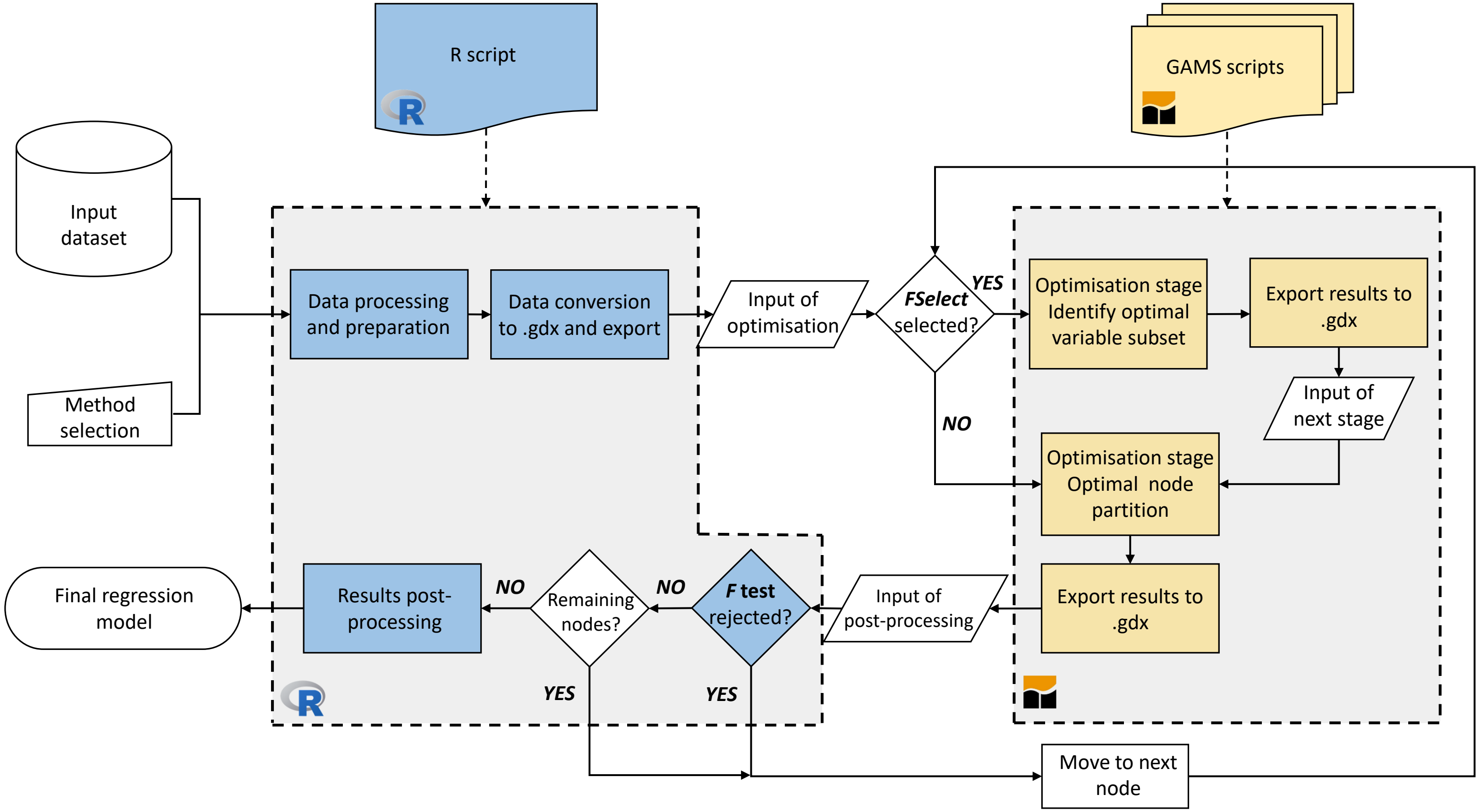


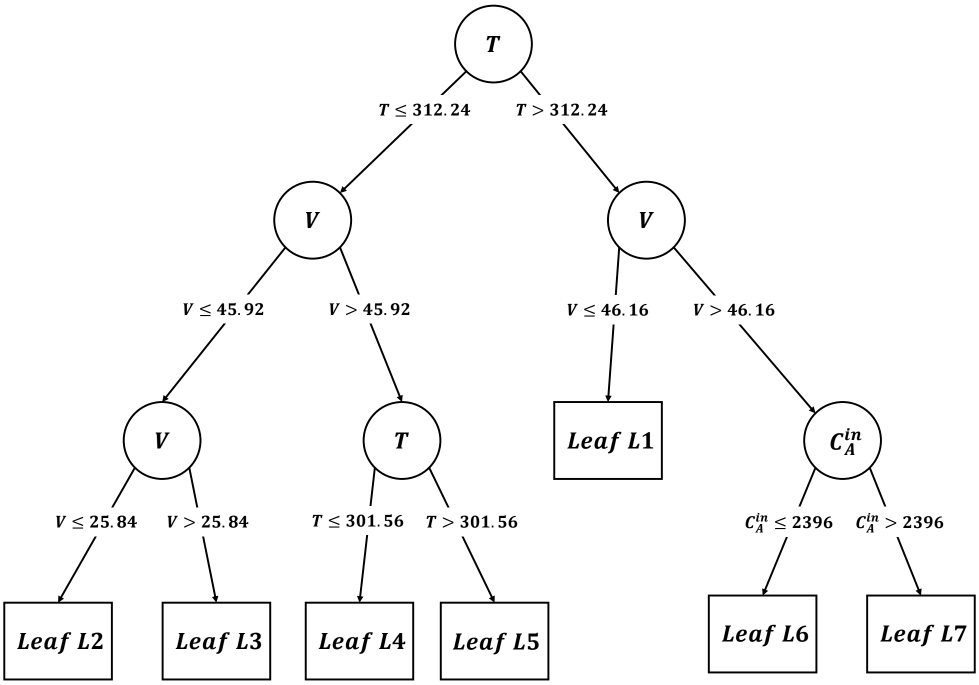
FIGURE 4.2: The implementation of the tree regression algorithm

4.3.2 Illustrative example

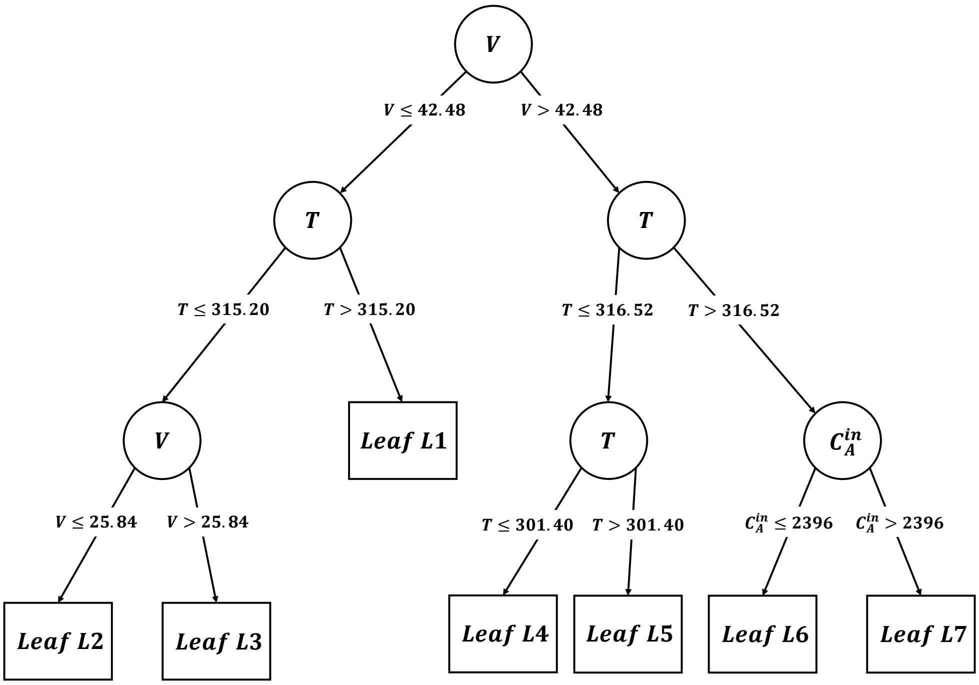
This example describes a continuous stirred tank reactor (CSTR), where a chain reaction takes place $A \rightarrow B \rightarrow C$. The dataset contains 4 input variables that describe the temperature of the reactor (**T**), concentration of reactant A ($\mathbf{C_A^{in}}$) and B ($\mathbf{C_B^{in}}$) in the inlet stream and the volume of the reactor (**V**). The output to be predicted is the production rate of reactant B (**Y**) (Palmer and Realff, 2002).

Figures 4.3a and 4.3b are the generated trees by applying the algorithms in Figures 4.1a and 4.1b respectively. For this example, the tree in Figure 4.3b was constructed by setting $N = 1$ as the maximum number of selected variables, showcasing the difference between the two extreme cases of exhaustive search and a greedy approach of selecting a single candidate variable.

The generated trees look similar, since both of them have 7 leaf nodes which translates to 7 regression models. However, the breaking rules are different. The main difference is that the root node is split on a different variable.



(A) Tree generated from StatTree algorithm as described in Figure 4.1a



(B) Tree generated from StatTree algorithm as described in Figure 4.1b

FIGURE 4.3: Regression trees of the CSTR example as generated by the proposed algorithms.

4.3.3 Datasets examined in this work

A number of examples are considered in this work which are summarised in table 4.1. Six new additional datasets are introduced in this chapter which include **abalone** and **speeding** that are available through the `datasets` package in `R`, **boston** is available at *StatLib* and **Dee**, **plastic** and **Wankara** are availabe through *KEEL* (Alcalá-Fdez et al., 2011).

TABLE 4.1: 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			

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 a given piece of plastic can 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.

4.3.4 Algorithm validation and comparison

For comparison purposes, various literature methods are implemented to test the performance against the proposed approaches. These methods include CART using the rpart package (Therneau et al., 2018), M5P regression using the RWeka package (Hornik et al., 2009; Witten et al., 2016), CTree using the partykit package (Hothorn and Zeileis, 2015) and MPtree which was implemented in R and GAMS.

Once again, 5-fold cross-validation is selected to evaluate the performance of the proposed algorithm. This procedure is repeated 10 times and the *MAE* is calculated. The final reported score is the average of all the runs.

All of the datasets examined in this work undergo feature scaling in the range of [0,1] for the reasons that have already been explained in Sections 2.3 and 3.3.4. Once again, feature scaling will make the selection of the parameters for the *bigM* constraints easier.

The confidence level of the StatTree algorithm is set to 99% for these computational runs. Such a selection means that the statistical test will be very strict in terms of rejecting the null hypothesis of the Chow test and generate new nodes.

The hyper-parameters of the competing decision tree regressors are set to their default values. These values are the ones proposed by the researchers that created the algorithms or the developers of the packages in the R programming language.

4.4 Computational results

4.4.1 Cross-validation runs

Table 4.2 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 the *MAE* metric for 7 out of 15 examples. Cubist is the next best performer with 3 out 15, MPtree and M5P both have 2 out 15 and CART only has a single dataset. However, that alone is not a good indication of overall performance.

TABLE 4.2: Cross-validation results 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. The procedure followed for the creation of this figure is the same as the one of Figure 3.5.

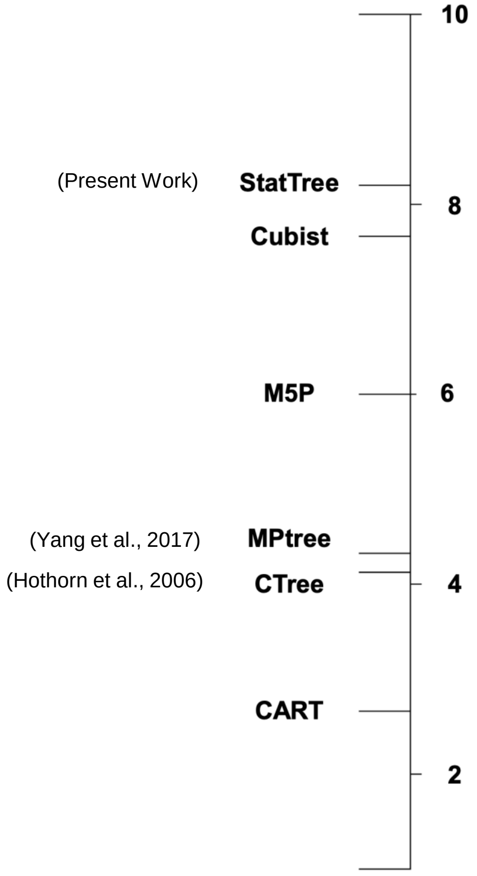


FIGURE 4.4: Visualisation of the performance of the methods based on the *MAE* results

Looking at Figure 4.4 makes it easier to compare the overall performance of the algorithms. A large performance gap exists between `StatTree` and `MPtree`, which indicates that the new proposed method improves upon the weaknesses of the previous one. `Cubist` 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 statistically significant difference in the results.

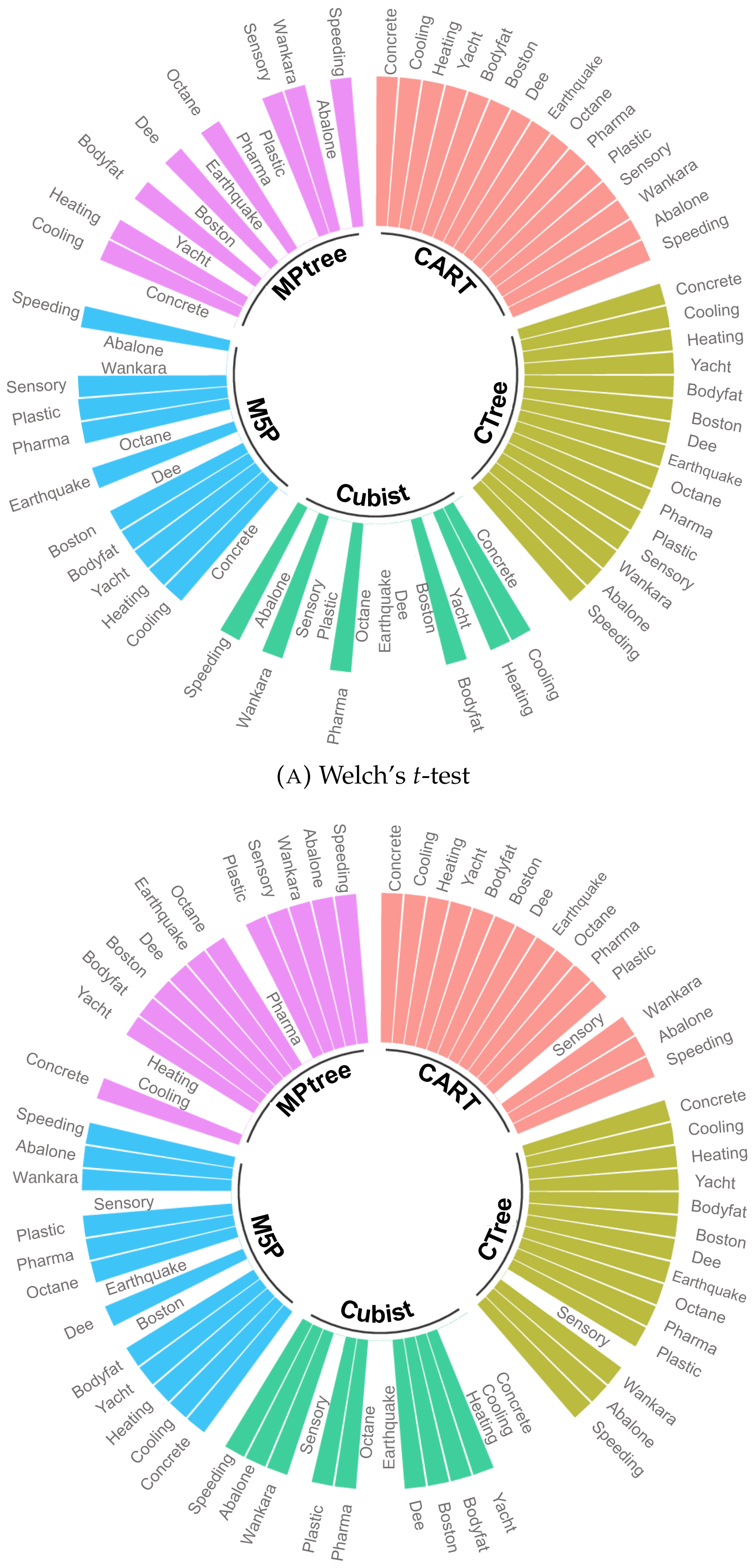
4.4.2 Statistical analysis

The same statistical testing procedure that was applied in Section 3.4.2 will be followed here as well. The Welch's *t*-test is well suited for checking the hypothesis that two populations have equal means when they have unequal variances. The same steps will be followed which include the calculation of the t-statistic using Equation B.1, selection of a confidence level of 99% $(\alpha = 0.01)$, calculation of the probability p-values of the t distribution and finally the rejection of the null hypothesis if $p < \alpha$.

Once again the two different populations will be the *CV* results of table 4.2 between `StatTree` and one of the other established regression algorithms. If there is a statistically significant difference in the results, then the method that achieved lower average error values will be considered the better one.

Figure 4.5 is a visual representation of the statistical analysis. 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 4.5a is a visualisation of the statistical analysis that has been performed to compare the methods. A bar is present only if there is a statistically significant 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 4.5b is similar to 4.5a, 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.



(A) Welch's *t*-test

(B) *MAE* scores

FIGURE 4.5: Visualisation of the computational results of this work. Case (a) is a representation of the statistical analysis using Welch's *t*-test. Case (b) is a representation of the *MAE* performance.

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, that translates to strong evidence of suggesting that the *MAE* averages of all the *CV* runs are indeed different and `StatTree` has provided better error values than the competitor.

Figure 4.6 is a representation of the performance of `StatTree`. Similarly to Section 3.3, each sub-figure contains two graphs. The outer ring represents the percentage of examples for which there is a statistically meaningful difference in the results. The pie chart in the middle represents the percentage of datasets for which `StatTree` is better than the competitor in terms of *MAE* score, provided that there is a statistical difference in the results.

`CTree` and `CART` have clearly underperformed compared to the proposed algorithm. There is a statistical difference in every single examined dataset and it is clear that `StatTree` has better performance, since it is better in 93% of the cases. `M5P` performs a bit better, but it is still not able to beat the work presented in this chapter.

Compared to `MPtree`, which shares the same optimisation model for splitting nodes, the results are indicative of the improvements of the new method. There is statistical difference in about half of the examined examples, which could be explained by the fact that some datasets are common in both works. However `StatTree` is better 75% of the time which is a significant improvement.

Finally, against `Cubist` there is a difference in only 40% of the examined datasets and a winning percentage of 67%. Both of these numbers are the lowest out of all the algorithms, but the proposed approach is still able to beat the competitor.

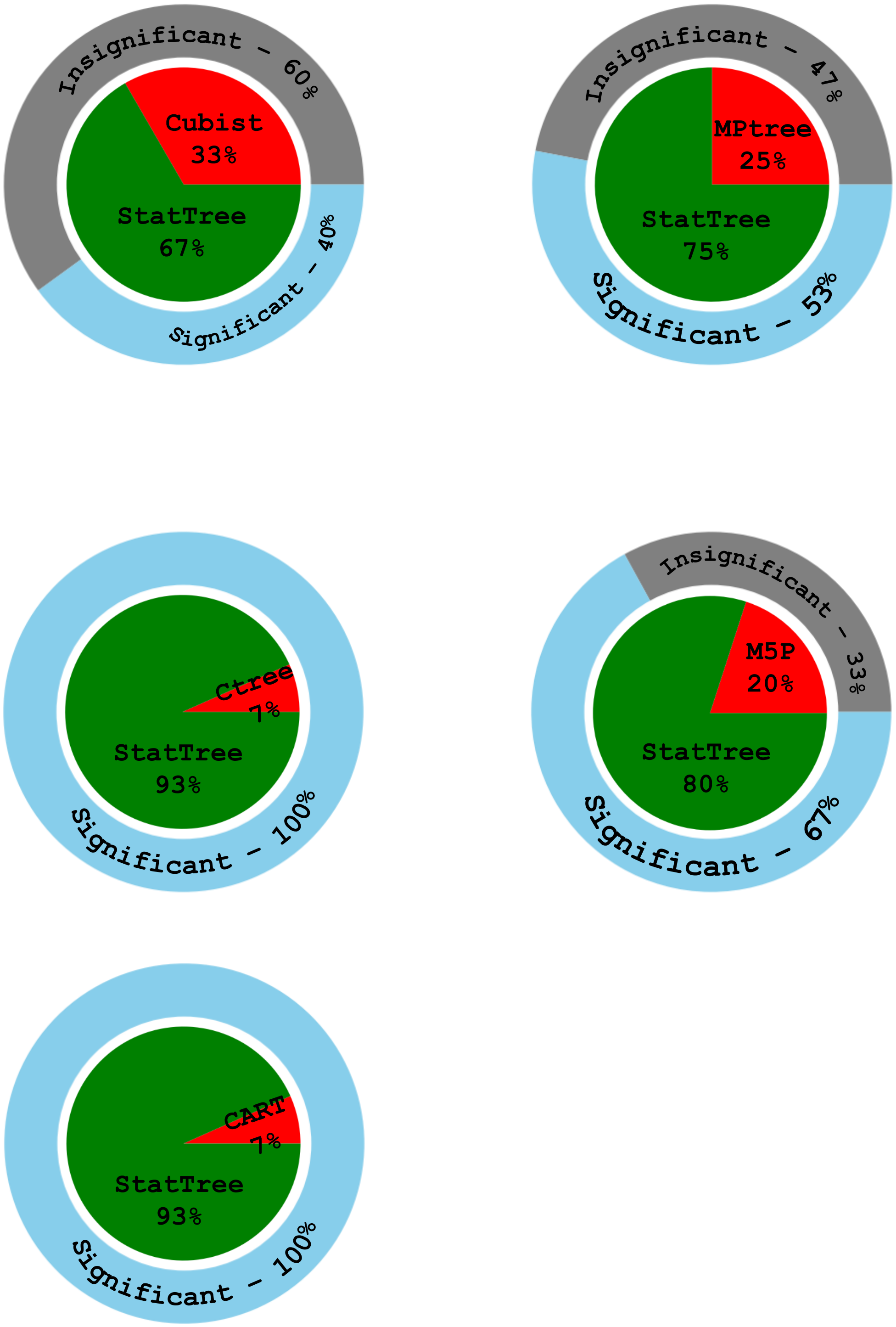


FIGURE 4.6: Percentage of winning between `StatTree` and the other methods, based only on the datasets with meaningful statistical difference.

4.4.3 Variable selection model

This section is dedicated to testing the performance of the algorithm of figure 4.1b. The inclusion of the `FSelect` model in the overall `StatTree` algorithm adds the benefit of reduced training time. However, there might be a compromise in performance, especially when the maximum number of selected features is set to low values.

In order to test how performance is affected, the same validation and statistical testing procedure will be followed for the same datasets. The comparison will be between `StatTree` – exhaustive search, `StatTree` – `Fselect` $N = 1$ and `StatTree` – `Fselect` $N = 3$. The value of $N = 1$ is chosen in order to demonstrate the extreme case where the user desires to obtain a single variable in the optimal subset. Such a value is expected to reduce computational expenses drastically but at the cost of accuracy. On the other hand, the value of $N = 3$ is expected to be somewhere in the middle when it comes to gains in computational time and sacrifices in accuracy, at least for the complexity of the examined datasets.

Higher values could have been tested for parameter N. However not all of the examined datasets in this work have large dimensionalities, which means that the value of parameter N in some cases would have been very close to the full input space and resemble exhaustive search.

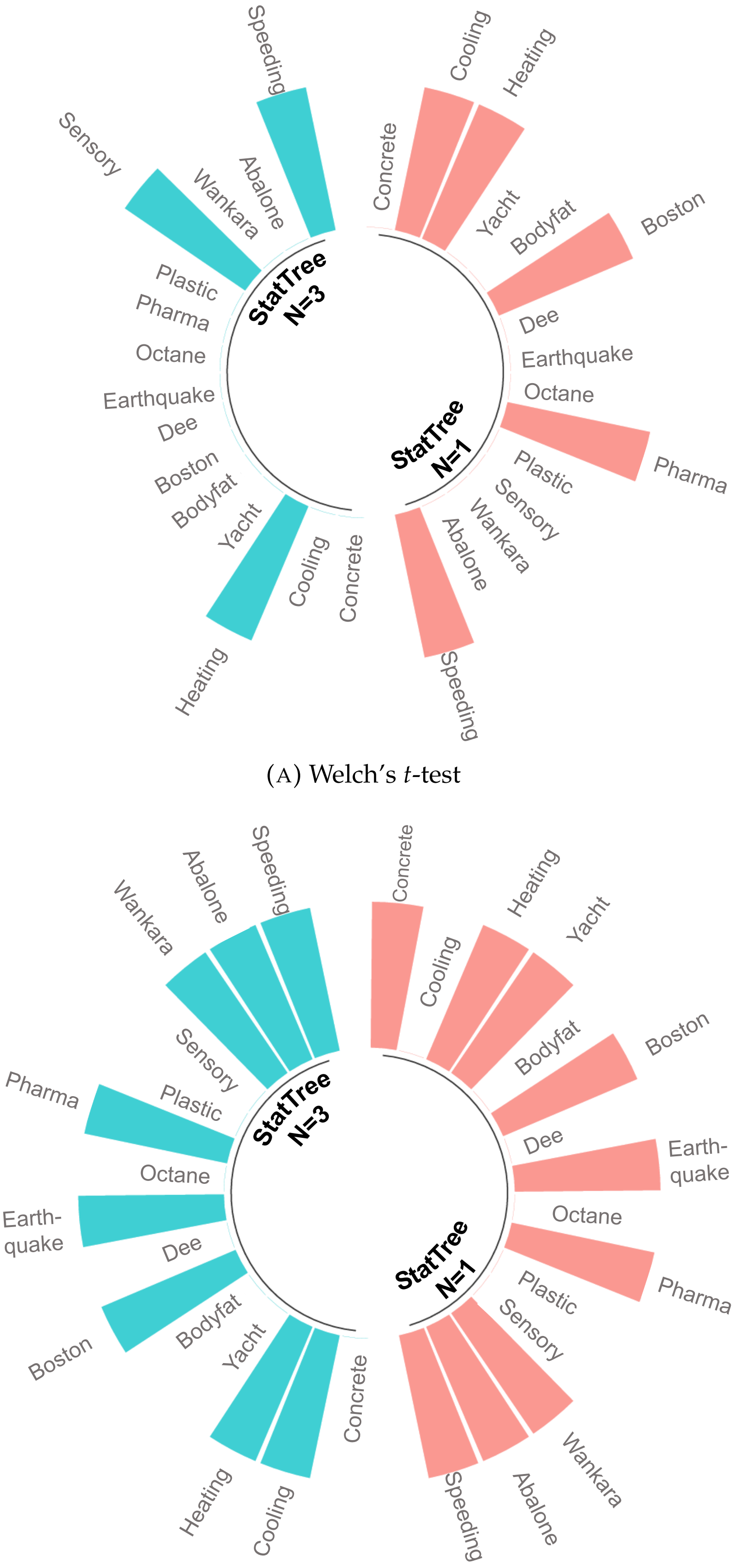
Table 4.3 contains the results of the 10 cross validation runs. The table captures the *MAE* metric for each dataset but also the total *CPU* time in seconds. It is apparent that the exhaustive search approach provides the best *MAE* scores for most of the examples. However, as in the previous sections, a statistical analysis should provide a better and more in-depth understanding of the results.

Figure 4.7 is a visualisation similar to Figures 4.5 and 3.6. The top part of the figure is the result of the statistical analysis while the bottom part is predictive performance using the *MAE* metric. The main algorithm that is represented in the circular barplots it the exhaustive search and it is compared against `StatTree` with the `FSelect` model.

TABLE 4.3: Comparison between exhaustive search and subset selection

	`StatTree` - Exhaustive Search		`StatTree` with `FSelect`			
			N=3		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

Note that even though there is a difference in the *MAE* results in favour of exhaustive search, the statistical analysis indicates that there is not enough evidence to assume that the difference is significant. Especially in the case of $N = 3$, there is a statistical difference in only 3 datasets while in the case of $N = 1$, as expected, there is a difference in 5 datasets. So, in the case where the user provides a dataset with a large number of features, the addition of the `FSelect` model provides an interesting alternative in order to reduce the *CPU* time of training a model without sacrificing predictive performance.



(A) Welch's *t*-test

(B) *MAE* scores

FIGURE 4.7: Visualisation of the computational results of this work. Case (a) is a representation of the statistical analysis using Welch's *t*-test. Case (b) is a representation of the *MAE* performance.

At this point, it is worth noting that the there is a small difference in the *MAE* results between `StatTree` and `StatTree` with `FSelect` for all the datasets except for one. The performance on the **Boston** dataset is significantly worse when using `FSelect`. However, it has a similar score for the different values of the *N* parameter.

A closer inspection at the generated trees showed that the introduction of the `FSelect` caused the root of the trees to be split on a different feature compared to exhaustive search. So, all the subsequent tree branches were very different compared to `StatTree`, hence the significant difference in performance. Luckily, this pattern of considerably poorer performance when using feature selection was not observed for the other datasets.

4.5 Concluding remarks

In this chapter, a novel decision tree regressor is developed. This regressor uses an existing mathematical formulation for splitting nodes into child nodes and introduces a statistical test to control the creation of new nodes. The introduction of the Chow statistical test is a vital addition for assessing the quality of splittings compared to the previous work that developed the optimisation model (Yang et al., 2017), which used a heuristic rule for this step.

In order to enhance the capabilities of the proposed algorithm and make its use more practical, a novel optimisation model is proposed that tackles the topic of variable selection. This model is used to choose a partitioning variable and speed up overall training time.

The novel algorithm is tested by performing 10 runs of 5-fold cross-validation across a range of publicly available datasets. A number of decision tree algorithms are used in order to compare and evaluate the performance of the proposed method. In addition to cross-validation, a statistical analysis is performed as well to check for significance in the results. The insights gained from the analysis suggest that the novel algorithm is an improvement over the original `MPtree` work and is also a very competitive alternative to other established methods. Finally, the use of the variable selection model has been proven to have an impact on computational time while preserving much of the predictive performance.

The next chapter introduces another decision tree algorithm. However, the supervised task it targets is classification instead of regression. For the purposes of this algorithm, a novel optimisation model is developed to assign samples into classes.

5 Development of decision tree classifier

This chapter deals with the development of a novel decision tree classifier. A novel mathematical model that applies a linear transformation to the data and then assigns samples into classes is developed. Various real-world examples are used to test and compare the performance of the algorithm against other classifiers.

5.1 Introduction

As mentioned in Section 2.1.1, decision trees are easy to understand and implement but also powerful for making predictions. As a consequence, tree models are also widely used for classification tasks as well. So, instead of predicting a continuous number, the models assign samples to classes.

The construction of trees still follows the same recursive procedure of splitting nodes into two subsets. This process is repeated until all the instances at a specific node have the same class assignment. Once this happens, the development of that part of the tree is terminated (Witten et al., 2016).

Deciding the partitioning variable for each node is still a very important step. In classification analysis, the goal is to create a measure to capture the purity of each node, hence selecting the splitting variable that leads to the purest child nodes. One such popular metric is the information gain, that was introduced in Section 2.1.1. This metric is based on the change of information entropy between two states. In decision trees, the two states can be viewed as the splitting of a node into two subsets. Depending on which variable is chosen for the partitioning, different information gain values are obtained. The one that returns the highest value is viewed as being the one that results in the most homogeneous branches.

To address overfitting, standard approaches are applied in order to control the tree generation process. These approaches include stopping criteria, such as metrics and statistical tests as the ones described in Chapters 3 and 4, heuristic rules like defining the minimum number of samples before splitting a node or the maximum number of samples per leaf node. Another common practice is pruning, which has been described in Chapter 2.1.2.

In this chapter, a novel decision tree classifier is proposed. This algorithm utilises a novel mathematical formulation that identifies the optimal splitting variable of a node, optimise the value of the break point and assign samples into classes while minimising the number of misclassified samples. To achieve this, the mathematical model first applies a linear transformation to the data to create a new pseudo-feature, based on which the classification task will take place. The algorithm uses this formulation in a recursive way in order to generate a tree structure.

5.2 Proposed algorithm

This section proposes a novel decision tree classifier which uses an *MILP* model to split samples into nodes and assign them to classes. The algorithm, from now on know as *Optimal Decision Tree Classifier* (`ODT`), accepts a multivariate dataset with a single multi-class output variable and numerical input variables and returns a decision tree in which each leaf node contains rules for classifying samples into class ranges.

A brief description of the proposed algorithm is presented first and then the detailed mathematical model. Grasping the core idea of how the method works will make it easier to understand the optimisation model that is the basis of the classifier.

5.2.1 Description of the algorithm

As with other classic decision tree approaches, this classifier employs recursive binary splitting for generating the tree. The main idea of the algorithm can be explained in three basic steps, even though the model achieves the results with a single iteration. For each node splitting, the optimisation model applies the following procedure:

1. First, the optimal partitioning variable is identified and the value of the break point is calculated. This is the essential step for generating new nodes. The membership of each sample, meaning whether a sample now belongs to the left or right child node, is also calculated.

2. The second step focuses on the child nodes. At each child, a linear expression is fitted to the data to create a new pseudo-feature. Despite sacrificing part of the interpretability of the final classification model, this step reduces the dimensionality of the input data down to a single variable and enables the model to capture part of the non-linear trends that might exist in the data.

   This step is similar to approaches such as Linear Discriminant Analysis (LDA) or Principal Component Analysis (PCA). Both of these approaches create new representations of the data by projecting them onto new axes. Similarly, the optimisation model that has been developed for the proposed algorithm projects the samples of each child node to a new axis, using a linear expression.

3. The third step is to assign samples to a class based on the new pseudo-feature that was generated in the previous step. This class assignment is achieved by introducing the idea of ranges. Each class in the dataset is represented by a specific range on the new axis. The bounds of the ranges are calculated by the optimisation model. If a sample falls in the wrong range, then it will be considered missclassified. The objective of the model is to minimise the number of missclassified samples.

Once the classifier has been trained, it is later presented with new data to make predictions. The samples first follow the path from the root node to the "correct" leaf node, then they go through the linear transformation and based on their value they can be assigned to a class. It is important to note that if there is a sample that after the transformation has a value that does not fall within a range, then it will be assigned to closest the range.

In order to better illustrate the proposed classification methodology, a simple example is provided in Figure 5.1. This example is a synthetic dataset that was created for illustration purposes only and has two input variables (x_1 and x_2), 50 samples and 3 classes (red, blue and green).

The top of the figure is a visualisation of the dataset. Note that the input variables have both been scaled to the range of [0,1]. The optimisation model will optimise simultaneously the splitting variable and the value of the break point. The resulting tree structure is presented next to the scatter plot and is a simple tree with two leaf nodes. The chosen variable by the model is x_1 at the value of 0.600.

A closer examination of each child node separately is needed to understand how the algorithm works. The left child contains samples that belong to the green and red classes. The linear expression that was applied by the model is capable of separating these two classes hence making it easy to visualise the bounds of the constructed ranges. The result is 0 misclassifications.

Things are easier at the right child with only a single class being present. The linear expression uses variable x_2 in order to generate values for the new pseudo-feature. The result is a single range representing the only available class.

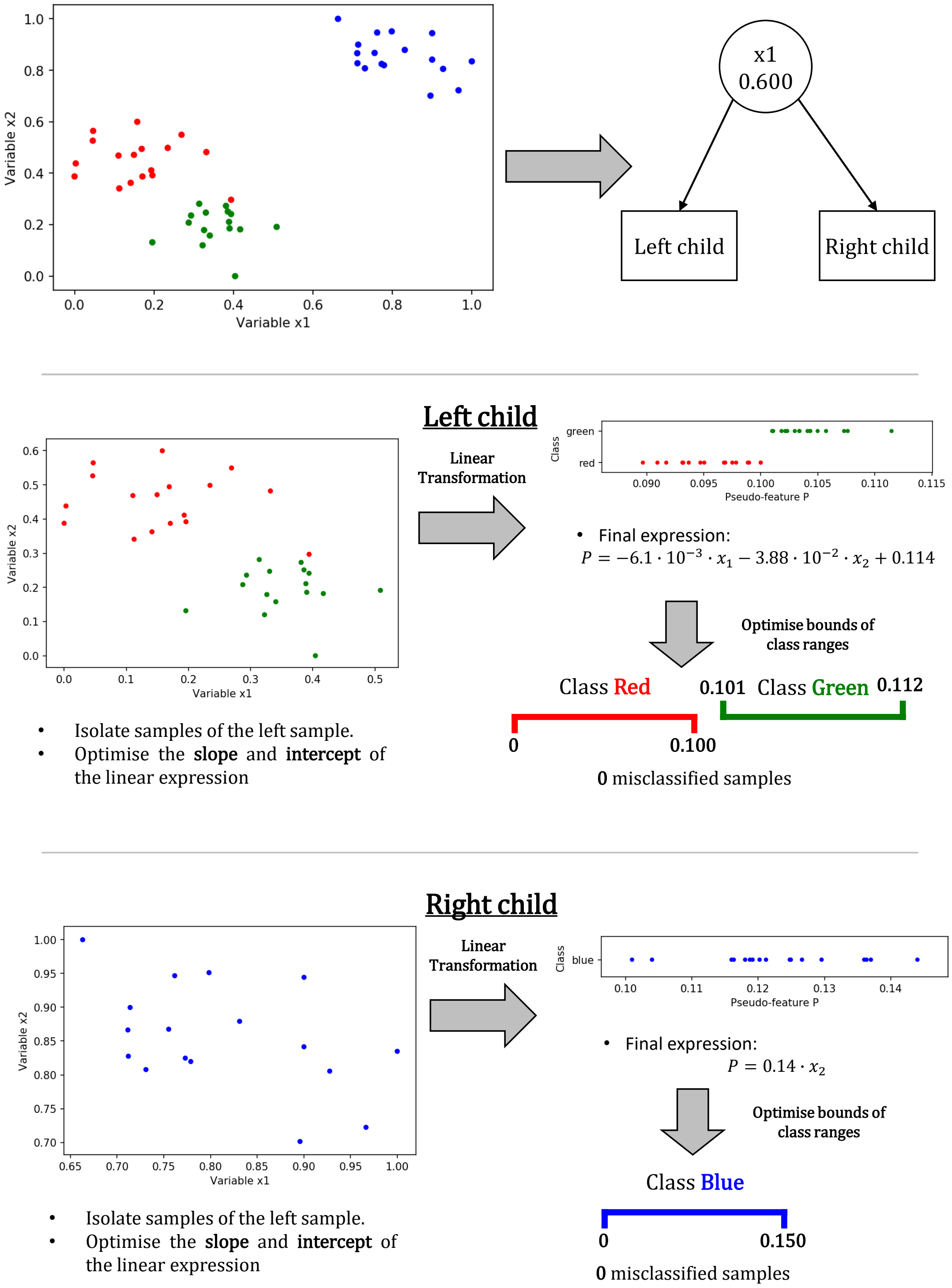


FIGURE 5.1: An example of the classification algorithm

5.2.2 Mathematical formulation of the optimisation model

The mathematical formulation of the model is presented below, along with the definition of all the indices, sets, parameters and variables used in the formulation

Indices

s	data samples, $s = 1, 2, ..., S$
m	features/input variables $m = 1, 2, ..., M$
c	child node of the current parent node; $c = left$ represents left child node, and $c = right$ represents right child node
g, k	available classes

Sets

Ω_g	set of samples that belong to class g

Parameters

A_{sm}	numeric value of sample s on feature m
U_1, U_2	suitably large positive numbers
ϵ	suitably small number

Variables

W_{mc}	regression coefficient for feature m in child node c
B_c	intercept of regression function in child node c

Positive variables

X_m	break-point value on feature m
P_{sc}	predicted output for sample s in child node c
$L_{g,c}$	lower bound of class g at child node c
$U_{g,c}$	upper bound of class g at child node c

Binary variables

F_{sc}	1 if sample s belongs to child c; 0 otherwise
Z_m	1 if variable m is selected as the partitioning variable; 0 otherwise
E_{sc}	1 if sample s is correctly classified at child c; 0 otherwise
Y_{gkc}	1 if class g is to the left of class k at child c; 0 otherwise
O_{gc}	1 if class g is present at child c; 0 otherwise

Mathematical constraints

The first step is to create a set of equations to split samples into two child nodes. To achieve this, a set of binary variables is used, F_{sc}, that allocates samples s to either the left or the right child node c. A new set of binary variables is introduced, Z_m, to capture the best partitioning variable. If for a specific input variable m the corresponding binary variable $Z_m = 1$ then this input variable is selected for the splitting, otherwise $Z_m = 0$.

Equation 5.1 assigns samples to the left child while Equation 5.2 assigns samples to the right child. Parameter ϵ is used to ensure that there will be a minimum difference between the break point values and the training samples.

$$A_{sm} \leq X_m + U_1 \cdot (2 - Z_m - F_{sc}) - \epsilon_1 \qquad \forall\, s, m, c = left \tag{5.1}$$

$$X_m - U_1 \cdot (2 - Z_m - F_{sc}) + \epsilon_1 \leq A_{sm} \qquad \forall\, s, m, c = right \tag{5.2}$$

The next two constraints formulate the assignment of samples to a single child node and the selection of the optimal partitioning variable. Equation 5.3 enforces that each sample s can be assigned to one child c, since the summation is in terms of the c index and the equation is generated for every sample s. Equation 5.4 chooses a single variable m as the partitioning feature since only one binary variable Z_m can take the value of 1.

$$\sum_c F_{sc} = 1 \qquad \forall s \tag{5.3}$$

$$\sum_m Z_m = 1 \tag{5.4}$$

Binary variables E_{sc} represent whether a sample s has been misclassified at child c. If $E_{sc} = 1$ then that sample is correctly classified. Equation 5.5 states that if a sample s does not belong to child c then that sample cannot be correctly classified at this child. A sample can be correctly classified at a child c only if it actually belongs to that child.

This is ensured by the following inequality. If $F_{sc} = 0$, then E_{sc} will be forced to the value of 0 since it is a binary variable. In contrast, if $F_{sc} = 1$, then E_{sc} is free to be {0,1}.

$$E_{sc} \leq F_{sc} \qquad \forall s, c \tag{5.5}$$

As already stated, this mathematical formulation applies a linear expression to the data, reducing the dimensionality to a single new pseudo-feature. Such an approach could potentially capture part of the complexities that exist in the data and improve predictive performance. The following constraint is a standard linear expression with slope coefficients W_{mc} and an intercept B_c.

$$Pr_{sc} = \sum_{m} A_{sm} \cdot W_{mc} + B_c \qquad \forall s, c \tag{5.6}$$

The following constraints assign samples to the correct class range. One of the tasks of the mathematical model is to partition the domain of the pseudo-feature of constraint 5.6 into ranges and each one will represent a single class. Every sample that falls within that range, gets the corresponding class assignment.

Variables L_{gc} and U_{gc} are the upper and lower bounds of the class ranges respectively. The *bigM* constraints 5.7 and 5.8 formulate that if sample s belongs to child c and is also correctly classified, then the pseudo-feature value P_{sc} should be within the upper and lower bounds of that class range. The supervised information is passed into these constraints through the Ω_g dynamic set, which maps the observed class assignment of each sample. As a result, the following set of equations is only generated when sample s belongs to class g.

$$Pr_{sc} \geq L_{gc} - U_2 \cdot (2 - E_{sc} - F_{sc}) \qquad \forall c, g, s \in \Omega_g \tag{5.7}$$

$$Pr_{sc} \leq U_{gc} + U_2 \cdot (2 - E_{sc} - F_{sc}) \qquad \forall c, g, s \in \Omega_g \tag{5.8}$$

Two new sets of binary variables are introduced in the formulation, Y_{gkc} and O_{gc}. These two represent the relative position of the classes based on the new pseudo-feature and the existence of a class in a child node respectively. Equations 5.9 and 5.10 formulate the position of the upper and lower bounds of all the available classes at a child c. These two equations examine the available classes in a pairwise fashion. To ensure that no combination of two classes will be repeated and examined twice, the constraints will be generated only when g>k. For example, checking the position of the bounds for classes A and B is exactly the same as checking the bounds for B and A and therefore it should not be checked twice.

$$U_{gc} + \epsilon_2 \leq L_{kc} + U_2 \cdot (3 - O_{gc} - O_{kc} - Y_{gkc}) \qquad \forall c, k, g > k \tag{5.9}$$

$$U_{kc} + \epsilon_2 \leq L_{gc} + U_2 \cdot (2 - O_{gc} - O_{kc} + Y_{gkc}) \qquad \forall c, k, g > k \tag{5.10}$$

Each child node will have a number of ranges representing the available classes. Hence, the model should enforce that if a class g is not present at child c, then the corresponding upper and lower bound values of the ranges should be zero. Otherwise, the model should be free to select any appropriate value. The following two constraints address this issue using *bigM* constraints. If the binary variable $O_{gc} = 1$, then class g is present at child c and therefore is multiplied with parameter U_2 to guarantee that the bounds are free to receive an appropriate value. On the other hand, if binary variable $O_{gc} = 0$, then class g is not present at child c and as a result the bounds are forced to zero.

$$L_{gc} \leq U_2 \cdot O_{gc} \qquad \forall g, c \tag{5.11}$$

$$U_{gc} \leq U_2 \cdot O_{gc} \qquad \forall g, c \tag{5.12}$$

The next constraint allows the user to select the number of classes that are allowed to be present at child c. Parameter N is user specified and can affect the computational time of training. The larger the value of this parameter, the more binary variables are introduced to the optimisation, making the model more expensive.

$$\sum_g O_{gc} \leq N \qquad \forall c \tag{5.13}$$

If a class g is present at child c, then the upper and lower bounds for that class should have different values, with the lower bound always being smaller than the upper bound. Parameter ϵ_3 controls the difference between the two bounds.

$$U_{gc} \geq L_{gc} + \epsilon_3 \cdot O_{gc} \qquad \forall g, c \tag{5.14}$$

Similarly to Equation 5.5, an additional constraint is required to check whether sample s has been correctly classified at child c. The added constraint formulates that the class of sample s has to be present at child c in order for that sample to be correctly classified. At Equation 5.15 that follows, if binary variable $O_{gc} = 1$, then class g is present at child c which enables the binary E_{sc} to be {0,1}. Otherwise, if $O_{gc} = 0$, then E_{sc} is forced to be zero. It is worth noting that this constraint is only generated for the elements of the Ω_g set which includes the supervised information of the classification task.

$$E_{sc} \leq O_{gc} \qquad \forall c, g, s \in \Omega_g \tag{5.15}$$

Finally, the objective function of this mathematical model is the minimisation of the total misclassified samples. A sample can be correctly classified in only one child. If that sample is correctly classified, then $E_{sc} = 1$, hence minimising the difference $1 - E_{sc}$.

$$\min \sum_s \left(1 - \sum_c E_{sc}\right) \tag{5.16}$$

5.3 Numerical and computational section

5.3.1 Algorithm implementation

For the implementation of this algorithm, the selected language was `python v.3.6` (Python Software Foundation , 2018) combined with `GAMS` for the optimisation parts. Python is a high-level, general-purpose programming language that has become very popular in the machine learning community due to its powerful libraries. Figure 5.2 illustrates the implementation of the algorithm.

The concept of calling `GAMS` using another programming language is present in this implementation as well. All the elements that are in blue boxes have been implemented in `python`, whereas the yellow ones in `GAMS`.

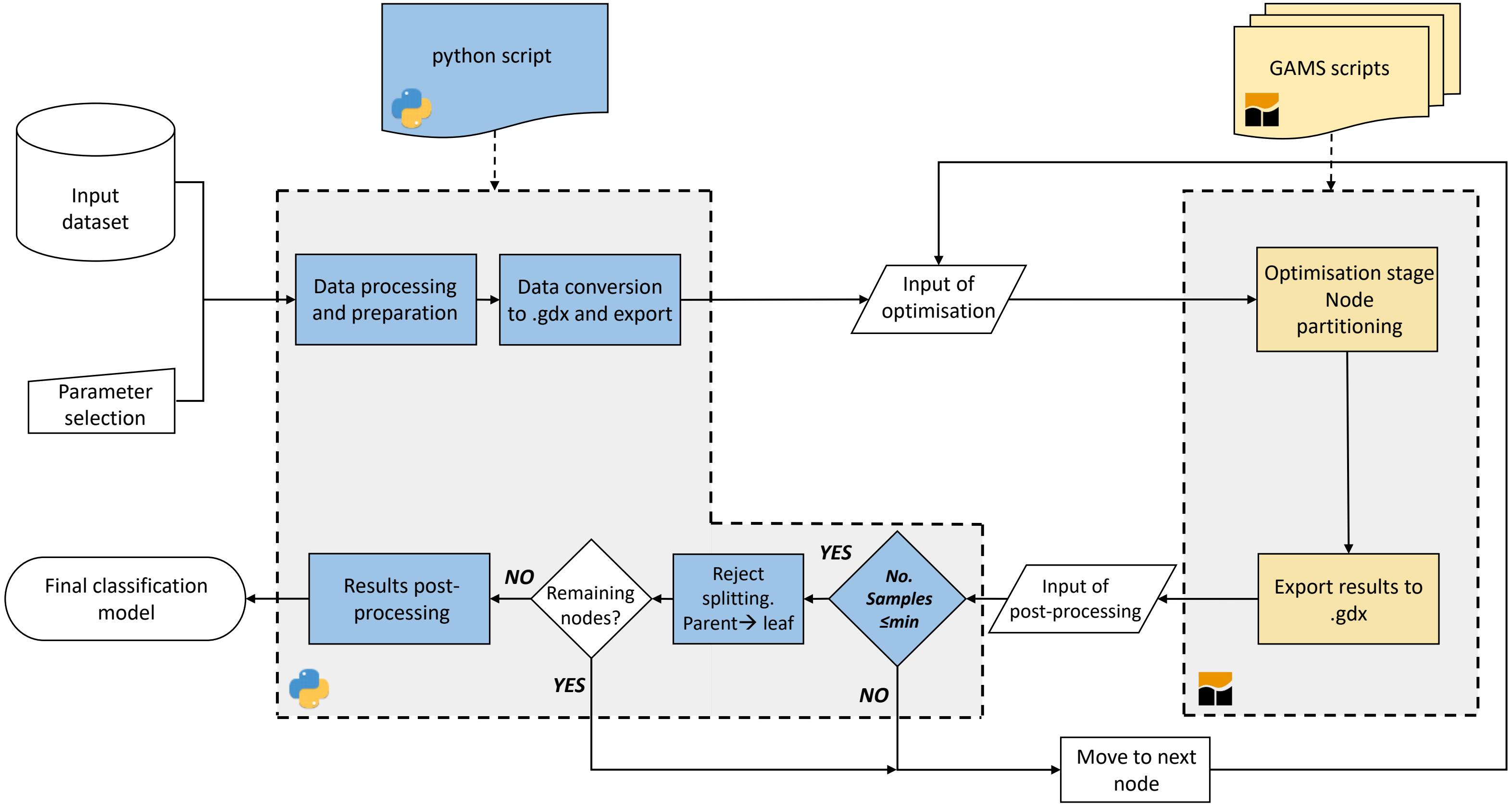


FIGURE 5.2: The implementation of the tree classification algorithm

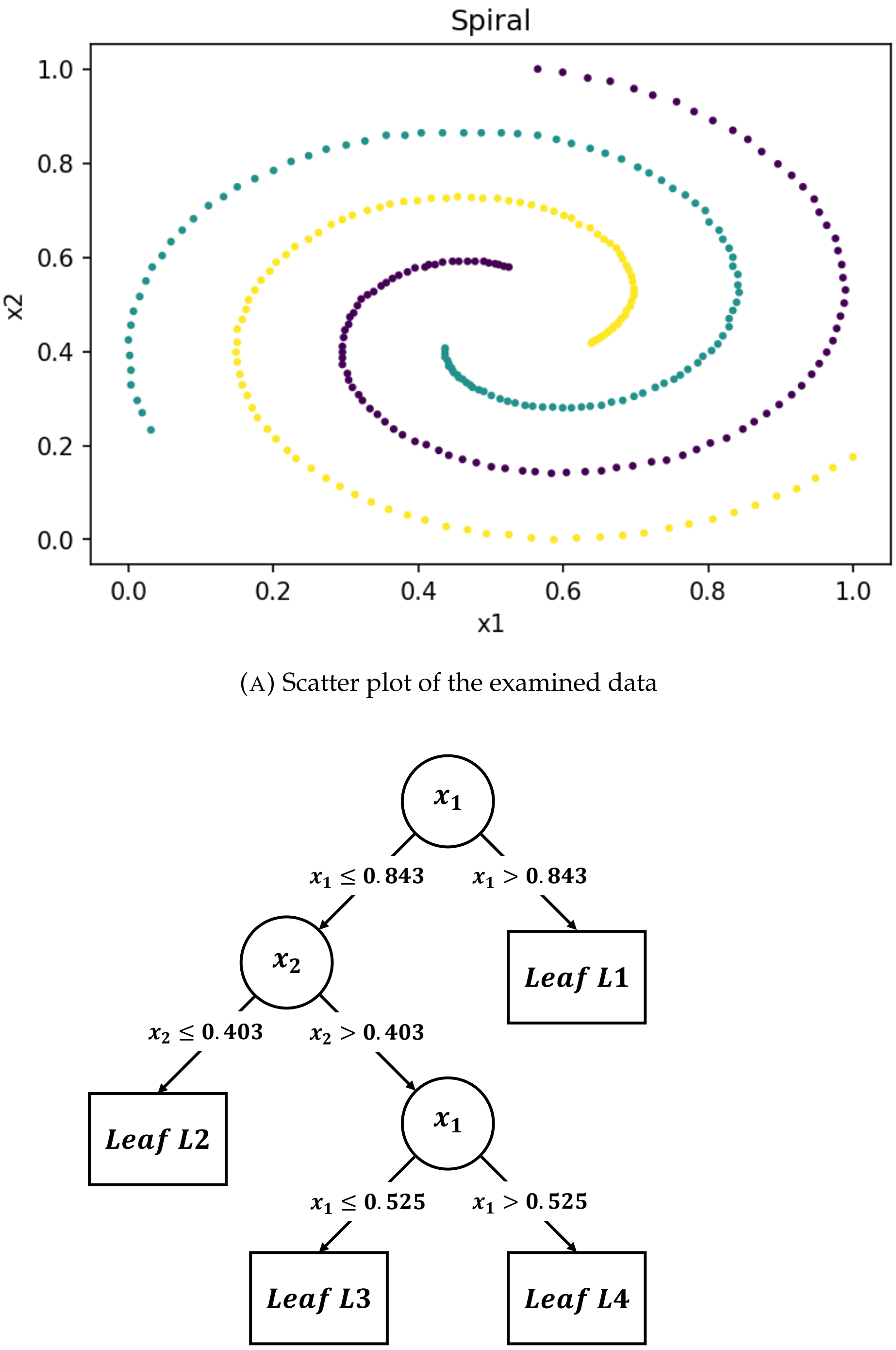
As an input, the user has to specify a parameter which controls the minimum number of samples allowed at each leaf node, as well as parameter N (defined at Equation 5.13) which controls the maximum number of allowable classes per node and provide the input dataset. The next step of the implementation is to process the data and transform them into the correct format. Once again, the format used for this implementation is `.gdx` and the conversion is achieved through a library called `gdxpds`.

The dataset is later exported using the `.gdx` format and it is imported into `GAMS`, where the optimisation model of Section 5.2.2 is executed. This optimisation model simultaneously identifies the partitioning variable and the value of the break point.

Once the model converges to a solution, the results are exported to `.gdx` and re-imported to `python` for further analysis. The `python` script checks whether the termination criterion has been satisfied and chooses which node should be examined next for further splitting, or if a node should be flagged as terminal and finish the generation process for that part of the tree branch. When the tree generation process stops, the final classification model is extracted. It is a tree structure where each leaf node contains the parameters of the linear transformation and the upper and lower bounds for each class.

The implementation of this algorithm follows the rules of the popular machine learning library `scikit-learn` (Pedregosa et al., 2011) for creating a custom estimator. `scikit-learn` is a python library that contains state-of-the-art machine learning algorithms for various tasks such as regression, classification and clustering, as well as functionalities such as model evaluation, grid search, pipelines etc. Creating a custom estimator by following the `scikit-learn` conventions and rules is very useful, since the resulting implementation will be compatible with all of the functionalities and utilities of the library, making it easier to apply various machine learning methodologies to the developer's custom work. One additional advantage is the fact that future users will be able to use this custom work without any extra training if they are already familiar with the `scikit-learn` library.

5.3.2 Illustrative example



(A) Scatter plot of the examined data

(B) Constructed tree

FIGURE 5.3: Illustrative example for the Optimal Decision Tree classifier. Figure (a) is a scatter plot of the examined dataset. Figure (b) is the constructed tree.

A synthetic dataset from the literature is used to demonstrate a classification model generated by ODT. This dataset is commonly used for both clustering and classification tasks and it consists of 3 classes that follow a spiral shape (Chang and Yeung, 2008). Figure 5.3a is an illustration of the dataset which has 312 samples, 2 input variables and each class is represented with a different colour.

Figure 5.3b is the tree structure that was generated by ODT. The resulting tree is relatively small with only 4 leaf nodes. Table 5.1 that follows summarises the rules of each leaf node that comprise the overall classification model.

TABLE 5.1: Final classification model using the ODT algorithm

Leaf	Transformation	Ranges
L1	$P_{L1} = 0.430 \cdot x_2$	purple: [0.101 – 0.366] yellow: [0.000 – 0.100]
L2	$P_{L2} = -0.306 \cdot x_1 - 0.654 \cdot x_2 + 0.840$	purple: [0.489 – 0.589] green: [0.000 – 0.488] yellow: [0.590 – 0.690]
L3	$P_{L3} = 0.464 \cdot x_1 - 0.513 \cdot x_2 + 0.320$	purple: [0.202 – 0.302] green: [0.000 – 0.100] yellow: [0.101 – 0.201]
L4	$P_{L4} = -0.568 \cdot x_1 - 0.386 \cdot x_2 + 0.823$	purple: [0.000 – 0.100] green: [0.101 – 0.201] yellow: [0.202 – 0.302]

The final classification model contains 7 nodes, 4 of which are terminal nodes. Each terminal node has a linear expression that transforms the multivariate data into a new pseudo-feature, called P. Furthermore, each leaf node contains the class ranges that have been determined by the optimisation model, based on that new feature.

Predicting the class of new samples involves the following:

- Assignment to correct leaf node, by following the path from the root node until a leaf node based on the break point values.
- Application of the linear transformation, and calculation of the new feature (P).

- Assignment to one of the class ranges. If the sample does not fall into any of the ranges, then the sample is assigned to the nearest range.

5.3.3 Datasets examined in this work

A number of examples are considered in this work which are summarised in table 5.2. Two datasets are synthetic and have been generated using software, whereas the rest are available through online sources. The *UCI* repository is the main online source of classification examples for this work.

TABLE 5.2: Classification datasets

Name	Samples	Variables	Classes
Gaussian	1000	2	2
Normally	1000	2	4
Aggregation	788	2	7
Compound	399	2	6
Firm_1	46	4	2
Firm_2	83	13	2
Iris	150	4	3
Pathbased	300	2	3
Toy	373	2	2
Sale	440	6	3
Wifi	2000	7	4
Banknote	1372	4	2
Patients	579	11	2
Modeling	403	5	4

The **Iris** dataset describes the morphologic variation of Iris flowers of three related species (Anderson, 1936; Fisher, 1936). The objective is to classify samples into one of the three available classes. The **Sale** dataset refers to clients of a wholesale distributor and includes the annual spending in monetary units on diverse product. The objective is to classify samples into one of three available classes that represent regions.

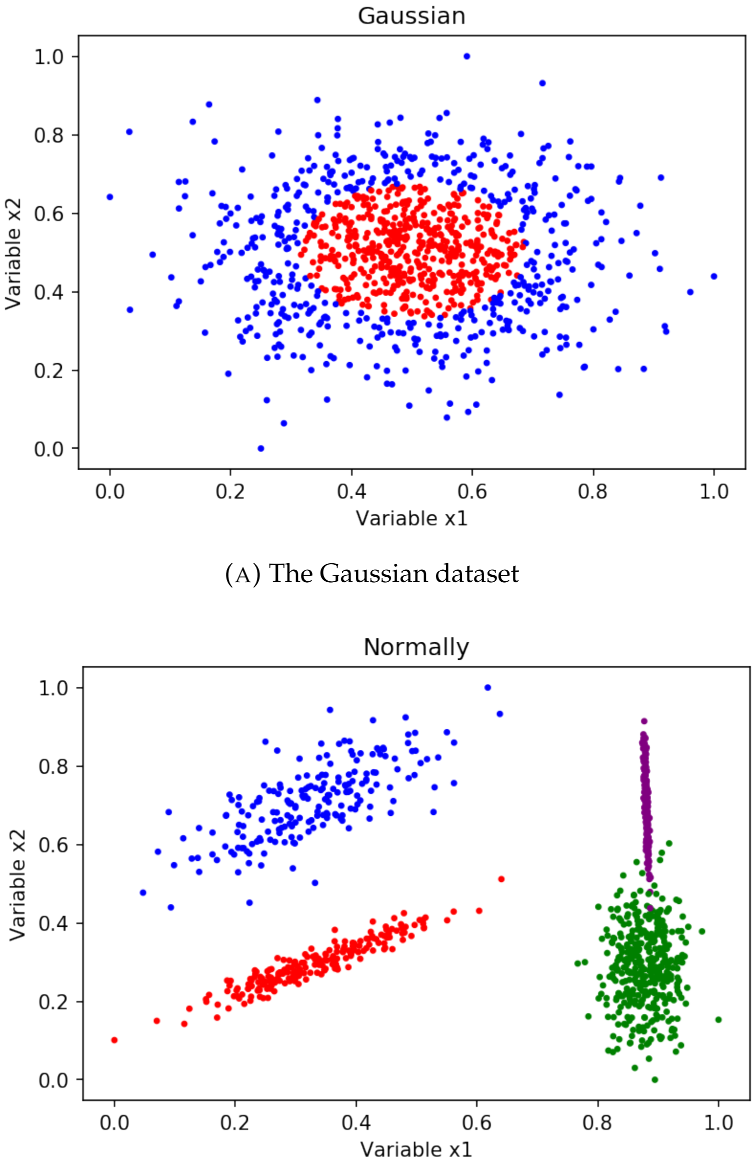
The **wifi** set contains data that were collected to perform experimentation on how wifi signal strength can be used for user localisation in an indoor environment. The **Patients** dataset contains data about liver patients. This is a binary classification problem with the target variable being positive or negative at being a liver patient (Ramana et al., 2012).

Two additional small examples are introduced in this work for comparison purposes. **Firm_1** consists of four financial rations and the target is either to classify samples as bankrupt or non-bankrupt firms (Nath and Jones, 1988). **Firm_2** is related to the corporate bankruptcy in US electric power industry and once again the target variable contains two classes (Sueyoshi, 2006).

A number of shape sets have been used in this work that are available online (Fränti and Sieranoja, 2019, 2018). These include the **Aggregation**, **Compound**, **Toy** and **Pathbased** datasets. Additionally, two more synthetic examples have been introduced to test the predictive performance of the proposed algorithm.

The **Gaussian** synthetic dataset, which was constructed with the use of `make_gaussian_quantiles` utility of the `scikit-learn` library, contains 1000 samples, 2 input variables and 2 classes and it was constructed by taking a multi-dimensional standard normal distribution and defining classes separated by nested concentric multi-dimensional spheres. Similarly, the **Normally** synthetic dataset was constructed by the `make_classification` utility, contains 1000 samples with 2 input variables and 2 classes, and it has clusters of normally distributed points.

Figure 5.4 that follows, contains the scatter plots of the two syntetic datasets in order to illustrate the degree of difficulty of the classification task. Figure 5.4a describes the **Gaussian** dataset. This dataset is designed to test the performance of the classification algorithm on non-linear data. On the other hand, Figure 5.4b will test the perfromance of the algorithm in the case of linear correlation in the data. The two datasets represent a more and less challenging task respectively.



(A) The Gaussian dataset

(B) The Normally dataset

FIGURE 5.4: Scatter plots of the constructed synthetic datasets using python and the `scikit-learn` library

5.3.4 Algorithm validation and comparison

Various classifiers are implemented in this work in order to compare the predictive performance of the novel proposed algorithm. These classifiers are all part of the `scikit-learn` library in python and include *Random Forest*, *Decision Trees* (`CART`), *Multi-layer Perceptron* (`MLP`), *Linear Discriminant Analysis* (`LDA`), *Quadratic*

Discriminant Analysis (QDA), and *Support Vector Machines* with both linear and radial basis function kernels (LSVM and RSVM respectively). Once again, 5-fold cross-validation is selected to evaluate the performance of the proposed algorithm. This procedure is repeated 10 times and the accuracy score is calculated for each fold. The final score is the average of all the runs.

All of the datasets examined in this work undergo feature scaling in the range of [0,1]. Once again, feature scaling will make the selection of the parameters for the *bigM* constraints easier.

The minimum number of samples per leaf node, is set to be the number of features of each examined dataset. Furthermore, parameter N is set to the value of $|g|/2$ of each examined dataset, rounding up when needed. Based on the notation of the mathematical formulation, $|g|$ is the cardinality of the g set and it represents the number of available classes. The exception to that rule are datasets that only have 2 classes.

The hyper-parameters of the competing classifiers are set to their default values. These values are the ones proposed by the developers of the library in python.

5.4 Computational results

5.4.1 Cross validation runs

Table 5.3 contains the accuracy scores of all the runs of cross validation and once again the method with the highest accuracy for each dataset is marked with bold. The proposed ODT algorithm provides competitive predictive performance for the examined case studies.

Examining the two constructed synthetic datasets, all of the methods provide similar accuracy scores, with ODT being the best performer on the **Normally** dataset and having good accuracy on the **Gaussian** dataset. It is worth noting that for the **Gaussian** dataset, ODT is able to perform better than *Decision Trees* (CART), which indicates that the linear transformation of the samples at each leaf node has an effect on performance.

TABLE 5.3: Classification performance of *CV* runs based on accuracy percentage (%)

	ODT	**RandFor**	**CART**	**MLP**
Gaussian	97.52	96.20	94.80	98.60
Normally	**99.50**	98.30	98.10	98.60
Aggregation	**100.00**	**100.00**	**100.00**	99.74
Compound	**97.00**	**97.00**	**97.00**	93.48
Firm_1	84.00	82.00	74.00	82.00
Firm_2	89.20	92.94	89.42	90.58
Iris	**99.30**	95.35	94.02	98.68
Pathbased	99.32	98.70	94.68	**100.00**
Toy	99.00	**99.75**	97.60	96.02
Sale	71.40	66.15	58.90	**72.30**
Wifi	97.10	**98.20**	96.78	97.10
Banknote	**99.80**	98.80	98.28	99.70
Patients	**73.00**	72.40	65.50	71.54
Modeling	**95.30**	90.80	87.90	94.30
	LDAC	**QDAC**	**LSVM**	**RSVM**
Gaussian	53.80	96.80	64.20	**98.80**
Normally	98.60	99.00	99.10	98.80
Aggregation	99.50	99.75	96.30	96.20
Compound	87.75	96.50	82.70	83.00
Firm_1	80.00	**90.00**	78.00	76.00
Firm_2	92.94	**96.46**	94.10	76.50
Iris	99.00	97.34	95.36	96.00
Pathbased	62.68	97.00	66.00	72.32
Toy	93.35	93.35	97.00	96.30
Sale	72.05	52.50	**72.30**	71.62
Wifi	97.08	97.00	97.00	97.70
Banknote	93.32	98.90	98.00	98.40
Patients	71.90	56.20	73.00	71.20
Modeling	93.30	93.80	90.65	93.82

Another interesting fact is the poor performance of all the algorithms for the **Sale** and **Patients** datasets. This level of performance can be attributed to the fact that there is a very poor correlation between input and output. Each feature is correlated less than 0.05 and 0.15 with the output of the **Sale** and **Patients** data respectively. These datasets were included to test how the various algorithms could handle such a task. For any practical application, these datasets would have to undergo feature engineering first, in order to improve classification performance.

ODT has the highest accuracy score for 7 of the examined examples, but again this table is not a clear comparison of all the classifiers. Similarly to Sections 3.4.1 and 4.4.1, a graph is developed in order to rank the overall performance of each algorithm. The procedure followed for the creation of this figure is the same as the one of Figure 3.5 and Figure 4.4.

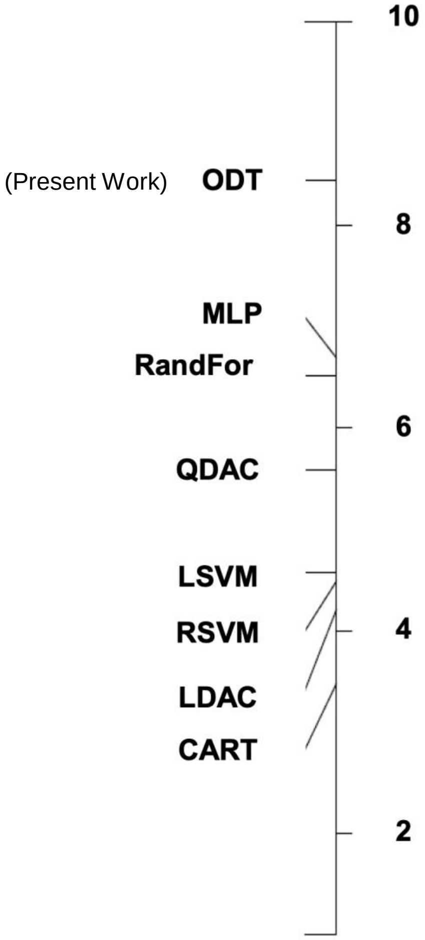


FIGURE 5.5: Visualisation of the performance of the methods based on the *MAE* results

Figure 5.5 is a visual aid of the overall comparison of the classifiers. This figure illustrates that there is a difference between standard *Decision Trees* and the proposed algorithm, whereas *Random Forest* and MLP are closer in terms of performance. *Random Forest* is expected to perform better than CART since it is an ensemble method with decision trees as the base classifier.

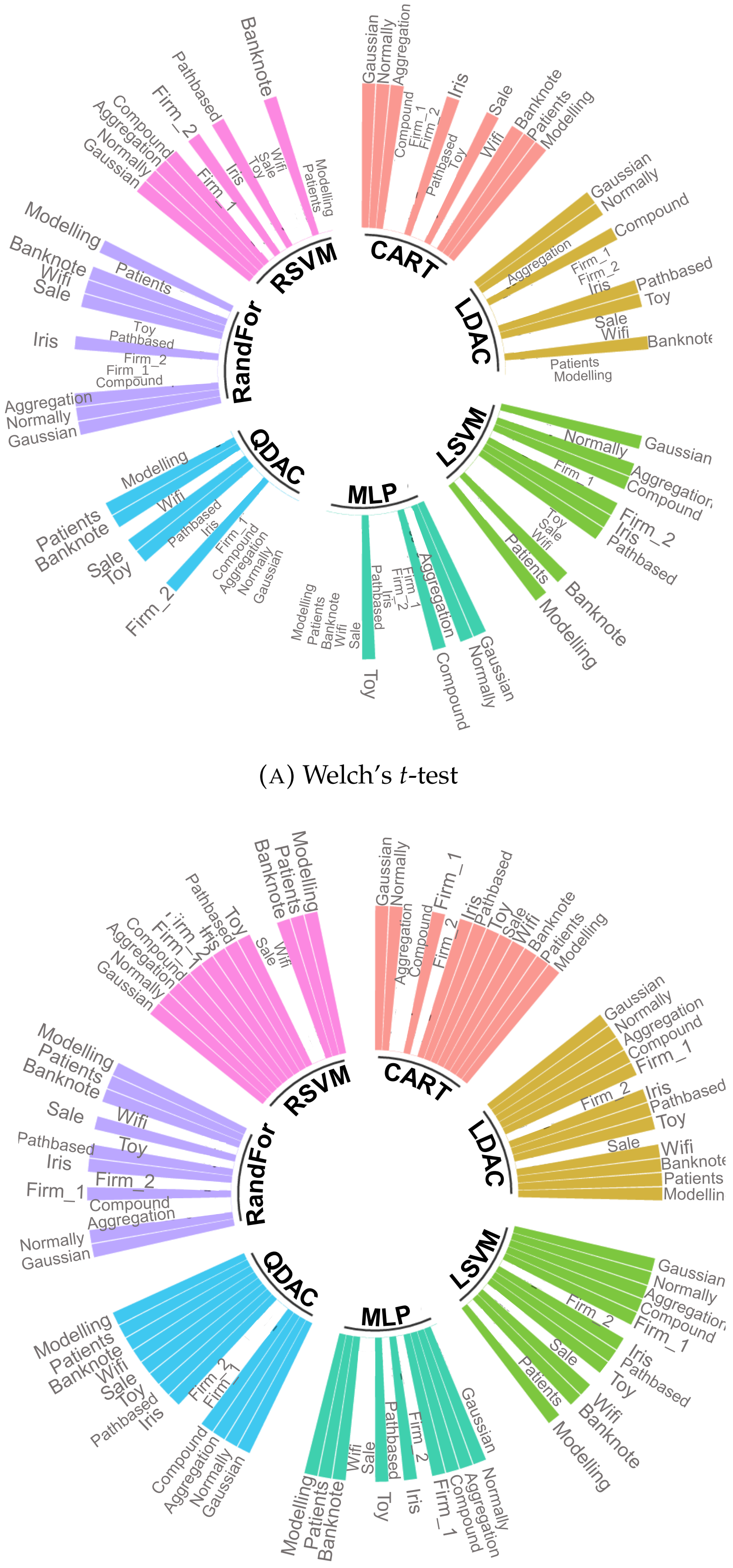
5.4.2 Statistical analysis

Figure 5.5 is not fully representative of the actual performance since many methods have similar performance with small differences in accuracy. Following the same approach as described in the previous chapters, the Welch's t-test will be applied to the results of the *CV* runs.

The two examined samples are the 10 runs of 5-fold cross-validation of the proposed ODT algorithm and one of the rest. Through this pairwise comparison, figure 5.6 is constructed which is a visual representation of the overall comparison. Sub-Figure 5.6a is a representation of the statistical analysis where a bar is present if there is a statistically meaningful difference in the means of the *CV* runs. There is a stastically meaningful difference between ODT and CART, RSVM, LSVM and *Random Forest* for most of the examples. This is not the case however with MLP and QDAC, with a statistical difference only for 4 and 5 examples respectivelly.

On the other hand, sub-Figure 5.6b is a representation of table 5.3 where a bar is present if ODT has a better accuracy score than the competing classifiers. For the most part, ODT is able to perform equally good or even better compared to the rest of the algorithms.

In order to compare the accuracy scores of all the classifiers, both sub-figures should be taken into account. If for a specific example there is a bar present in both figures then this translates to a statistically meaningful difference in the accuracy scores of the *CV* runs, with ODT providing better performance. Otherwise, the competing classifier is better, or there is not enough evidence to declare a winner.



(A) Welch’s *t*-test

(B) Accuracy scores

FIGURE 5.6: Visualisation of the computational results of this work. Case (a) is a representation of the statistical analysis using Welch’s *t*-test. Case (b) is a representation of the accuracy score.

Based on that statement, there is no clear winner compared to `MLP`, `LDAC` and `QDAC`. Against these classifiers, there is no evidence of statistical difference in the results for 10, 8 and 9 datasets respectively. So, even though for the rest of the examples `ODT` is able to outperform the competitors, most of the times the results are so similar that it is hard to come to a conclusion.

Against *Random Forest*, the proposed algorithm performed better for 6 datasets while losing in 2, with the rest not having a statistically meaningful difference. Compared to `LDAC`, the null hypothesis of the statistical analysis was not rejected for 8 examples. Despite that, for the rest 6 datasets the proposed classifier is the winner. Similar results were obtained against `RSVM` where `ODT` performs better for half of the examined examples.

Finally, the same results are acquired against `CART` and `LSVM`, with `ODT` winning in 7 examples, losing in 1 and not providing statistically different accuracy scores for the rest 6 of the examples.

5.5 Concluding remarks

This chapter introduces a decision tree classifier that is based on a novel mathematical formulation. This formulation is an *MILP* optimisation model that is responsible for splitting a node into two subsets. The model can identify the partitioning feature for the splitting and perform a linear transformation to the data by optimising the linear coefficient terms. The objective function of the mathematical model is to minimise the number of misclassified samples.

For the evaluation of the algorithm, various real world and synthetic datasets are used. Once again, 10 runs of 5-fold cross-validation are performed as well as a statistical analysis. Popular algorithms from the literature are used to compare the performance of the proposed algorithm. The results of the analysis indicate that the algorithm is able to compete and provide good accuracy scores against other methods.

The next chapter contains some concluding remarks about this thesis. It also highlights the main contributions of this work and includes suggestions about potential future steps.

6 Concluding remarks

This chapter summarises the main findings of this doctoral thesis. Furthermore, in this section some possible directions for future work are outlined.

a

The work presented in this thesis is concerned with developing machine learning algorithms using mathematical programming models and mixed integer optimisation. This study involves the development of supervised learning algorithms that can deal with regression and classification tasks and provide new alternatives to the ever-growing library of machine learning and data mining algorithms.

Overall, this work is characterised by several proposed algorithms that extend and add novel pieces to the existing literature. These features include the addition of model selection metrics to a current algorithm in order to extend its mathematical formulation and improve its predictive accuracy and the introduction of statistical testing to a tree regression algorithm to control the tree generation process. Other novelties include a subset selection optimisation model to reduce the input space when searching for a partitioning variable for a tree regression algorithm and a new mathematical formulation for generating decision trees for classification.

All of the new additions and the novel mathematical formulations that were developed are described in detail in separate chapters of this thesis. A summary of the main contributions and findings of this work is discussed in the following sections.

6.1 Development of a piecewise regression algorithm

In this work, a novel piecewise regression algorithm that employs mathematical optimisation models to perform the analysis was proposed. The work extended the model of the existing `OPLRA` algorithm and proposed four different variants.

The existing OPLRA algorithm used an *MILP* model to segment the dataset into regions and then a heuristic iterative approach in order to select the optimal number of regions for the regression model. This heuristic approach introduced a new parameter, called β, as a threshold to the reduction percentage of the absolute error while adding regions.

To eliminate this parameter, this work proposed the use of the *Akaike* and the *Bayesian Information Criteria*. These two criteria are well established for model selection tasks and were used to select the final number of regions. To achieve this, two different approaches were proposed for the new variants. The first approach employed an iterative procedure, where multiple optimisation models were solved until the optimal number of regions was identified. After each iteration, the values of the information criteria were calculated by post-processing the optimisation results. The second approach employed a single-level *MILP* strategy, where a single model was solved and the objective function of the model was the actual criterion. In this approach, an upped bound for the maximum number of regions was provided and then the model decided on the optimal solution.

To test the accuracy of the new approaches and the potential improvements over the OPLRA algorithm, a total of 10 real-world datasets were used. The evaluation of the model was achieved by performing 10 runs of 5-fold cross-validation on all the datasets. The results were compared to other regression algorithms from the literature. Based on the results of this work, the variant called PROA outperformed many of the competing regression algorithms for half of the examined datasets and also achieved competitive performance for the remaining examples.

A statistical analysis was performed on the results in order to check for statistical significance. The analysis determined that the reported difference in the MAE scores between PROA and the other algorithms was proven to be statistically significant in most cases. It was noted that the *BIC*-based methods could not perform as well as PROA.

It was therefore concluded that the single-level *MILP* variant called PROA was a good regression alternative that had competitive performance against other established methods as it outperformed them in some examined datasets.

However, despite the competitive performance, one of the key limitations of this algorithm was the inability to handle big datasets with a large number of samples. But for relatively smaller datasets that are similar to the ones examined in this work, the algorithm was performing well, while retaining interpretability and an easy to understand model.

6.2 Development of a decision tree regressor

The work in Chapter 4 addressed the issue of multivariate regression analysis by generating tree structures. In previous work, a regression tree algorithm was developed called `MPtree`, which constructed tree structures using an *MILP* model and then assessed the quality of the partitions based on a heuristic. This heuristic rule introduced a new parameter that was used as a threshold in the reduction of the error of each new split compared to the error on the root node.

A new approach was proposed that used the same optimisation model to split data into nodes, but introduced the well established 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. The objective function of the model was 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 was compared to other established tree regression algorithms that were 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 had an impact on the generated trees.

In addition to the core algorithm, a novel mathematical model 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 was 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, it was demonstrated that there was not a big compromise in predictive performance, since the results were very similar and in most cases the differences between them were proven to be insignificant.

6.3 Development of a decision tree classifier

This work concerned the development of a novel classification algorithm that generated decision trees. This novel algorithm used an optimisation model in order to split nodes into two subsets. At each new child node, the model fitted a linear expression to the samples of that node and generated a new pseudo-feature. Based on the values of this new pseudo-feature, the model created ranges of samples and each range was assigned to a class. Every sample that fell into that range was assigned to that class. The objective of the optimisation model was to minimise the number of misclassified samples.

Several examples were used to test the performance of this classification algorithm. Those examples included both synthetic and real-world examples in order to cover a variety of classification tasks. The selected validation approach was to perform 10 runs of 5-fold cross-validation. The computational results demonstrated that the proposed algorithm was able to achieve classification performance that was very similar to other established classifiers and in many cases better than the competitors. However, the statistical analysis showed that there was no statically significant difference when compared to two of the seven competing classifiers.

Despite this fact, the proposed method was proven to be a good alternative classification algorithm that generated compact tree structures due to the added step of the linear transformation of the data at the leaf nodes. Computational complexity still remained an issue since the algorithm was not able to handle a large number of training samples.

6.4 Summary of main contributions

The main contributions of this work are:

- A piecewise regression algorithm that included model selection criteria to select the optimal number of regions. The novel variations of this algorithm included the post-processing of the optimisation results to include the criteria, as well as an extension of an existing mathematical model that was able to directly optimise the criteria and decide on the optimal number of regions.
- The introduction of a well-established statistical test to address the issues of an existing tree regression algorithm. This statistical test assessed the quality of splitting nodes hence providing a way of terminating the generation of new nodes.
- An additional subset selection mathematical model that selected an optimal subset of variables of the original input space. This model was combined with the tree regression algorithm in order to speed up the process of generating new nodes, by eliminating the previous approach of exhaustive search over the entire set of variables.
- A novel mathematical model for generating decision tree classifiers. This model identified the optimal partitioning variable, applied a linear expression to the data and assigned them to "ranges" that represented classes. The model minimised the number of misclassified samples.

6.5 Directions for future work

6.5.1 Extending this research

In this work, mathematical models and algorithms concerning supervised machine learning tasks were developed. These tasks include regression (Chapters 3 and 4) and classification (Chapter 5). These algorithms were tested using real world datasets form online sources and compared to other established algorithms. In this last section, a few recommendations and items for future work are discussed that can potentially improve the algorithms both in terms of performance and computational speed.

- The piecewise regression algorithm of Chapter 3 is based on the partitioning of the input dataset into multiple segments and the fitting of a linear expression to each one. The resulting model is a collection of disjoint linear expressions, that are characterised by a number of break points.

  A possible future direction lies in the subject of continuity. There might be a case study in which ensuring continuity between the different linear segments is important. In recent work, Kong and Maravelias (2020) proposed mixed-integer programming models for fitting univariate discrete data points with continuous piecewise linear functions. Extending the proposed algorithm to ensure continuity while handling multivariate datsets, is an interesting topic for further research.

- Chapter 4 proposes a decision tree regressor. This algorithm uses an *MILP* model for splitting nodes and a statistical test check for significance and control the tree generation process. This algorithm follows the "conventional" way of splitting nodes, which means that each binary splitting rule is characterised by the selection of a single input variable. That variable can different at each node, but it is always a single variable.

  Research has been done to expand this idea and incorporate splitting rules that are based on multivariate selection instead of a single variable (Bertsimas and Dunn, 2017). The resulting tree is a collection of nodes where each node contains a rule that is a function of the input variables. A multivariate function can describe the input space better than a single feature and thus capture the behaviour of the data. Such an approach can potentially lead to improvement in predictive performance.

  A similar approach might worth investigating for the proposed algorithm. By modifying the constraints of the optimisation model in Section 4.2.1, it is possible to create multivariate splitting rules. These rules and modifications would also eliminate the exhaustive search approach for identifying the best partitioning variable, as well as the need of the variable selection model that was developed in Section 4.2.3.

However, having a function in the place of single variable rule has its drawbacks. One of the key advantages of decision trees is model interpretability. The inclusion of multivariate function rules adds an extra step of complexity and therefore creates tree structures which are more difficult to understand. Another drawback is added model complexity, and which leads to longer training times. New optimisation variables will be introduced to the optimisation model that affect both the computational time and the convergence of the model to a solution.

- Finally, a novel classification algorithm was developed in Chapter 5. This decision tree classifier minimises the number of misclassified samples by applying a linear transformation to the data. Similarly to Chapter 4, the resulting tree nodes contain binary splitting rules considering only a single input feature. In this formulation, the model decides on the best partitioning feature without the need of exhaustive search. The selection of the partitioning feature comes at the cost of computational time, due to the inclusion of the binary variables needed for this task. Moreover, this approach can lead to potential equivalent solutions. That means that two or more input features can lead to the same number of misclassified samples.

  A possible future direction could be the investigation of a way to select the best partitioning feature. Two of the most popular and well-established metrics for this task are the information gain and the gini impurity (Quinlan, 1986). These two metrics have been extensively used by other algorithms with success. However, novel mathematical models could be developed to include these metrics in an optimisation framework.

  These mathematical models could be used in order to determine the best partitioning feature at each node. Then the resulting information would be used as input to the existing mathematical model of Chapter 5.2.2, hence eliminating the need for the additional binary variables and making the optimisation model less expensive. Another implementation could be the inclusion of such metrics to the optimisation model of Section 5.2.2 directly. Instead of having a two-step approach of feature selection first and then the splitting of the nodes, this time the optimisation model would identify the partitioning variable, apply the linear transformation and classify the samples by directly optimising one of the metrics.

The main drawback of these approaches would be the non-linear nature of the metrics. The resulting models would be *MINLP* formulations. Throughout the development of the algorithms in this thesis, it was always desirable to have linear formulations that can be solved to global optimality with the available solvers. An *MINLP* formulation would add extra complexity that would have to be addressed.

6.5.2 Broader recommendations

The models and algorithms developed in this research work, which are based on the premise of data segmentation to enhance performance, enable the creation of supervised learning models that are useful for capturing the underlying mechanism that might exist in labelled datasets. It would be very interesting to use the propsed algorithms as black-box models to chemical engineering problems. An example was presented in Section 4.3.2, where the proposed decision tree regressor was used to describe the operation of CSTR reactor. This example, albeit small, demonstrated the ability to use this method as a way to create a surrogate model to describe a physical system. Future work could be targeted at finding additional case studies and test the performance of the algorithms to real world chemical engineering problems

Another direction could be the use of these models in data-driven approaches for model identification in kinetics. In recent work, Quaglio et al. (2020) combined supervised learning (training a neural network) to recognise kinetic models from experimental data. Specifically, the authors created synthetic data by using a library of possible kinetic models and generated random parameters for these models. For each kinetic model, the simulation procedure was repeated multiple times using different parameters, in order to generate a labelled dataset. A classifier was trained on the generated dataset to learn the physical properties of the system. Instead of training a neural network which is hard to interpret, it would be very interesting to apply the proposed decision tree classifier to such a case study. The generated decision tree could also provide insights about the kinetic models, by visualising the results.

However, for any practical application, the issue of dimensionality would have to be addressed as the current proposed methodologies can handle only a certain number of samples and variables. This is a common issue with integer programming models, which can be become hard to solve when the number of decision variables becomes large. This poses a problem since in the real world, the available datasets would be much larger.

This issue of dimensionality however, was partially addressed by Cardoso-Silva et al. (2019). The authors used the base mathematical model of the `OPLRA` algorithm and extended it to incorporate feature selection with regularisation. They applied their method to Quantitative Structure-Activity Relationship (*QSAR*) models, where the datasets typically involve a large number of features. A similar approach could be incorporated to the proposed algorithms in this work to handle slightly larger datasets, with the goal of applying this work to real chemical engineering problems.

A Chow test

The Chow test is a statistical test to check whether the coefficients in two linear regression models on different data sets are equal (Chow, 1960). It is most commonly used to test for the presence of a structural break in the data. In order to use the F-distribution to apply the statistical test, the first step is to compute the Chow F-statistic using equation 4.8. The equation is also presented in this appendix:

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

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 total number of samples in the dataset

The degrees of freedom for this statistical test can be computed by the two following equations:

$$p_2 - p_1 \tag{A.2}$$

$$n - p_2 \tag{A.3}$$

where:

p_1	regression parameters of model 1
p_2	regression parameters of model 2
n	total number of samples in the dataset

Once the F-statistic and the degrees of freedom have been computed, the F-distribution can be used to test the null hypothesis.

B Welch's t-test

The Welch's t-test is a two-sample test which is used to check the hypothesis that two populations have the equal means. In statistics, this test is also known as unequal variances t-test due to the fact that it is designed for samples with unequal variances (Welch, 1947). As stated in section 3.4.2, the procedure for applying the test includes calculating the t-statistic and then either the p-values or the critical t-value from the t-distribution table. The t-statistic is formulated as follows (Ruxton, 2006):

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}} \tag{B.1}$$

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

Calculating the p-values/t-critical value requires computing the degrees of freedom associated with the variance estimate first. The formula for the degrees of freedom is as follows (Ruxton, 2006):

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

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-distribution can be used to test the null hypothesis using a two-tailed test.

Publications that have arisen from this work

- Gkioulekas, I. and Papageorgiou, L.G. (2020). Tree regression models using statistical testing and mixed integer programming. Under review
- Gkioulekas, I. and Papageorgiou, L. G. (2019). Optimal regression tree models through mixed integer programming. *Data Science - Analytics and Applications*, 57 - 62.
- Gkioulekas, I. and Papageorgiou, L.G. (2019). Piecewise regression analysis through information criteria using mathematical programming. *Expert Systems with Applications*, 121: 362 - 372
- Gkioulekas, I. and Papageorgiou, L.G. (2018). Piecewise regression analysis through the Akaike information criterion using mathematical programming. *IFAC-PapersOnLine*, 51: 730 - 735.

Conference participation

- 2^{nd} International Data Science Conference, *iDSC* 2019. Salzburg, Austria.
  Oral presentation: Optimal regression tree models through mixed integer programming.
- ChemEngDay UK. 2019. Edinburgh, UK.
  Oral presentation: Optimal regression tree models through mixed integer programming. Awarded prize for best oral presentation in the Modelling and Process Systems.
- *IFAC* Symposium on System Identification, *SYSID* 2018. Stockholm, Sweden.
  Oral presentation: Piecewise Regression Analysis through Information Criteria using Mathematical Programming.

Bibliography

Acevedo, J. and Pistikopoulos, E. N. (1998). Stochastic optimization based algorithms for process synthesis under uncertainty. *Computers & Chemical Engineering*, 22:647–671.

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.

Anderson, E. (1936). The species problem in Iris. *Annals of the Missouri Botanical Garden*, 23:457–509.

Barber, D. (2012). *Bayesian reasoning and machine learning*. Cambridge University Press.

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.

Beygelzimer, A., Kakadet, S., and Langford, J. (2013). Package FNN. *Available at* `https://cran.r-project.org/web/packages/FNN/FNN.pdf`.

Bixby, R. E. (2012). A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, pages 107–121.

Breiman, L. (1996). Bias, variance, and arcing classifiers. Technical report, Tech. Rep. 460, Statistics Department, University of California, Berkeley.

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.

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

Cardoso-Silva, J., Papadatos, G., Papageorgiou, L. G., and Tsoka, S. (2019). Optimal piecewise linear regression algorithm for qsar modelling. *Molecular informatics*, 38:1800028.

Carrasco, O. C. (2019). Support vector machines for classification. *Available at* `https://towardsdatascience.com/support-vector-machines-for-classification-fc7c1565e3`.

Cartwright, H. (2020). *Machine Learning in Chemistry*. Issn Series. Royal Society of Chemistry.

Chaffart, D. and Ricardez-Sandoval, L. A. (2018). Optimization and control of a thin film growth process: A hybrid first principles/artificial neural network based multiscale modelling approach. *Computers & Chemical Engineering*, 119:465–479.

Chang, H. and Yeung, D.-Y. (2008). Robust path-based spectral clustering. *Pattern Recognition*, 41:191–203.

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.

Chetouani, Y. et al. (2007). Using artificial neural networks for the modelling of a distillation column. *IJCSA*, 4:119–133.

Chow, G. C. (1960). Tests of equality between sets of coefficients in two linear regressions. *Econometrica: Journal of the Econometric Society*, pages 591–605.

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, pages 603–619.

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

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., and Reis, J. (2009). Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47:547–553.

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.

dos Santos, M. T., Vianna Jr, A. S., and Le Roux, G. A. (2018). Programming skills in the industry 4.0: are chemical engineering students able to face new problems? *Education for Chemical Engineers*, 22:69–76.

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

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Fabozzi, F. J., Focardi, S. M., Rachev, S. T., and Arshanapalli, B. G. (2014). *The Basics of Financial Econometrics: Tools, Concepts, and Asset Management Applications*. John Wiley & Sons.

Fisher, A. R. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188.

Fortmann-Roe, S. (2012). Understanding the bias-variance tradeoff. *Available at* `http://scott.fortmann-roe.com/docs/BiasVariance.html`.

Fränti, P. and Sieranoja, S. (2018). K-means properties on six clustering benchmark datasets. *Applied Intelligence*, 48:4743–4759.

Fränti, P. and Sieranoja, S. (2019). Clustering basic benchmark. *Available at* `http://cs.joensuu.fi/sipu/datasets/`.

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.

Gevrey, M., Dimopoulos, I., and Lek, S. (2003). Review and comparison of methods to study the contribution of variables in artificial neural network models. *Ecological modelling*, 160:249–264.

Grus, J. (2019). *Data science from scratch: first principles with python*. O'Reilly Media.

Gurobi Optimization, LLC (2019). Gurobi optimizer reference manual. *Available at* `https://www.gurobi.com/products/gurobi-optimizer/`.

Hastie, T., Tibshirani, R., and Friedman, J. (2008). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, 2 edition.

Hawkins, D. M. (2004). The problem of overfitting. *Journal of Chemical Information and Computer Sciences*, 44:1–12.

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

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24:417.

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.

IBM (2019). IBM ILOG CPLEX Optimizer. *Available at* `https://www.ibm.com/analytics/cplex-optimizer`.

International Prognostic Factors Study Group (2010). Prognostic factors in patients with metastatic germ cell tumors who experienced treatment failure with cisplatin-based first-line chemotherapy. *Journal of Clinical Oncology*, 28:4906–4911.

Jain, R. and Dirkse, S. (2018). Package gdxrrw v.1.0.4. *Available at* `https://https://support.gams.com/gdxrrw:interfacing_gams_and_r`.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning with applications in R*, volume 112. Springer.

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.

Kellner, T. (2015). Deep machine learning: Ge and bp will connect thousands of subsea oil wells to the industrial internet. Technical report, GE Reports.

Khayet, M. and Cojocaru, C. (2012). Artificial neural network modeling and optimization of desalination by air gap membrane distillation. *Separation and Purification Technology*, 86:171–182.

Kimura, K. and Waki, H. (2018). Minimization of akaike's information criterion in linear regression analysis via mixed integer nonlinear program. *Optimization Methods and Software*, 33:633–649.

Kochenderfer, M. J. (2015). *Decision making under uncertainty: theory and application*. MIT press.

Kong, L. and Maravelias, C. T. (2020). On the derivation of continuous piecewise linear approximating functions. *INFORMS Journal on Computing*.

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243.

Kriegel, H.-P., Kröger, P., Sander, J., and Zimek, A. (2011). Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1:231–240.

Lehmann, R. and Lösler, M. (2016). Multiple outlier detection: Hypothesis tests versus model selection by information criteria. *Journal of Surveying Engineering*, 142:04016017.

Li, Z. and Ierapetritou, M. (2008). Process scheduling under uncertainty: Review and challenges. *Computers & Chemical Engineering*, 32:715–727.

Liaw, A. and Wiener, M. (2002). Classification and Regression by randomForest. *R News*, 2:18–22.

Liu, M. L. and Sahinidis, N. V. (1996). Optimization in process planning under uncertainty. *Industrial & Engineering Chemistry Research*, 35:4154–4165.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Maimon, O. and Rokach, L. (2005). *Data mining and knowledge discovery handbook*. Springer, 3 edition.

Malash, G. F. and El-Khaiary, M. I. (2010). Piecewise linear regression: A statistical method for the analysis of experimental adsorption data by the intraparticle-diffusion models. *Chemical Engineering Journal*, 163:256–263.

Medford, A. J., Kunz, M. R., Ewing, S. M., Borders, T., and Fushimi, R. (2018). Extracting knowledge from data through catalysis informatics. *ACS Catalysis*, 8:7403–7429.

Medina-González, S., Gkioulekas, I., Dua, V., and Papageorgiou, L. G. (2020). A graph theory approach for scenario aggregation for stochastic optimisation. *Computers & Chemical Engineering*, page 106810.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Friedrich, L. (2017). Package e1071. *Available at* `https://cran.r-project.org/web/packages/e1071/e1071.pdf`.

Milborrow, S. (2018). Package earth. *Available at* `https://cran.r-project.org/web/packages/earth/earth.pdf`.

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

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of Machine Learning*. MIT Press.

Montgomery, D. C., Peck, E. A., and Vining, G. G. (2012). *Introduction to linear regression analysis*, volume 821. John Wiley & Sons.

Muggeo, V., Sottile, G., and Porcu, M. (2020). Modelling COVID-19 outbreak: segmented regression to assess lockdown effectiveness. Technical report.

Muggeo, V. M. R. (2003). Estimating regression models with unknown breakpoints. *Statistics in Medicine*, 22:3055–3071.

Muggeo, V. M. R. (2008). Segmented:An R Package to Fit Regression Models with Broken-Line Relationships. *R news*, 8:20–25.

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

Nath, R. and Jones, T. W. (1988). A variable selection criterion in the linear programming approaches to discriminant analysis. *Decision Sciences*, 19:554–563.

Ning, C. and You, F. (2018). Data-driven decision making under uncertainty integrating robust optimization with principal component analysis and kernel smoothing methods. *Computers & Chemical Engineering*, 112:190–210.

Ning, C. and You, F. (2019). Optimization under uncertainty in the era of big data and deep learning: When machine learning meets mathematical programming. *Computers & Chemical Engineering*, 125:434–448.

Palmer, K. and Realff, M. (2002). Metamodeling approach to optimization of steady-state flowsheet simulations: Model generation. *Chemical Engineering Research and Design*, 80:760–772.

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

Petsagkourakis, P., Sandoval, I. O., Bradford, E., Zhang, D., and Chanona, E. A. d. R. (2020). Constrained reinforcement learning for dynamic optimization under uncertainty. *arXiv preprint arXiv:2006.02750*.

Psichogios, D. C. and Ungar, L. H. (1992). A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 38:1499–1511.

Python Software Foundation (2018). *Python Language, version 3.6.5*.

Quaglio, M., Roberts, L., Jaapar, M. S. B., Fraga, E. S., Dua, V., and Galvanin, F. (2020). An artificial neural network approach to recognise kinetic models from experimental data. *Computers & Chemical Engineering*, 135:106759.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1:81–106.

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.

Ramana, B. V., Babu, M. S. P., and Venkateswarlu, N. (2012). A critical comparative study of liver patients from USA and India: an exploratory analysis. *International Journal of Computer Science Issues (IJCSI)*, 9:506–516.

Rulequest (2020). Data Mining with Cubist. *Available at* `https://rulequest.com/cubist-info.html`.

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.

Sato, T., Takano, Y., Miyashiro, R., and Yoshise, A. (2016). Feature subset selection for logistic regression via mixed integer optimization. *Computational Optimization and Applications*, 64:865–880.

Shah, H. and Gopal, M. (2016). Model-free predictive control of nonlinear processes based on reinforcement learning. *IFAC-PapersOnLine*, 49:89–94.

Shang, C., Huang, X., and You, F. (2017). Data-driven robust optimization based on kernel learning. *Computers & Chemical Engineering*, 106:464–479.

Shannon, C. E. (2001). A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5:3–55.

Snipes, M. and Taylor, D. C. (2014). Model selection and akaike information criteria: An example from wine ratings and prices. *Wine Economics and Policy*, 3:3–9.

Sueyoshi, T. (2006). DEA-discriminant analysis: Methodological comparison among eight discriminant analysis approaches. *European Journal of Operational Research*, 169:247–272.

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

Thompson, M. L. and Kramer, M. A. (1994). Modeling chemical processes using prior knowledge and neural networks. *AIChE Journal*, 40:1328–1340.

Toms, J. D. and Lesperance, M. L. (2003). Piecewise regression: a tool for identifying ecological thresholds. *Ecology*, 84:2034–2041.

Tran, K. and Ulissi, Z. W. (2018). Active learning across intermetallics to guide discovery of electrocatalysts for co 2 reduction and h 2 evolution. *Nature Catalysis*, 1:696–703.

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.

Vasily Zubarev (2018). Machine learning for everyone. *Available at* `https://vas3k.com/blog/machine_learning/`.

Venkatasubramanian, V. (2019). The promise of artificial intelligence in chemical engineering: Is it here, finally? *AIChE Journal*, 65:466–478.

Venkatasubramanian, V., Chan, K., and Caruthers, J. M. (1994). Computer-aided molecular design using genetic algorithms. *Computers & Chemical Engineering*, 18:833–844.

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, Y. and Witten, I. H. (1996). Induction of model trees for predicting continuous classes. In *Proceedings of Poster Papers, Ninth European Conference on Machine Learning*.

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.

Wood, F. S. (1973). The use of individual effects and residuals in fitting equations to data. *Technometrics*, 15:677–695.

Xu, D. and Tian, Y. (2015). A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2:165–193.

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