FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

An optimization-based wrapper approach for utility-based data mining

José Francisco Cagigal da Silva Gomes

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Carlos Manuel Milheiro de Oliveira Pinto Soares

Co-Supervisor: Yassine Baghoussi

September 6, 2019

An optimization-based wrapper approach for utility-based data mining

José Francisco Cagigal da Silva Gomes

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Doctor Rui Carlos Camacho de Sousa Ferreira da Silva
External Examiner: Doctor Nuno Miguel Pereira Moniz
Supervisor: Doctor Carlos Manuel Milheiro de Oliveira Pinto Soares
September 6, 2019

Abstract

Traditional data mining focuses on finding patterns that are more frequently observed in data. However, there are tasks where rarer instances can be translated into high utility, such as fraud and environmental catastrophe detection. In real-world scenarios, decision making is often influenced by those regions of the data space, which are not too populated.

Standard machine learning algorithms usually focus on commonly used measures, such as root mean squared error, mean squared error and mean absolute error (for regression problems). These measures lead to models which are likely to make large errors on rarer instances. Therefore, these measures fall short when high accuracy is particularly important in specific areas of the data set, such as the ones with less density. Utility-based data mining assume non-uniform costs in data. As such, these approaches could be used on the aforementioned problems by developing utility functions that give higher importance to rare instances rather than common ones.

Most of the work on utility-based data mining focuses on classification problems. Thus, we concentrate on regression problems. In this dissertation, we propose a utility-based data mining wrapper approach that can be used on cost-sensitive tasks. The idea is to iteratively change the weight of the training set according to a given utility measurement. As this approach is domain independent, it can be used for regression and classification tasks.

To test our approach, we used artificial and UCI data sets. Results show that in 100% of the tests performed, it was possible to have a higher utility when compared to the machine learning algorithm using the original training set. We also developed a domain-specific utility measure to test this approach on a retail case study. The goal, in this case, was to predict daily net sales using the created metric. We obtained a higher utility score in all stores that were analyzed.

Keywords: Data Mining, Utility Based Data Mining, Utility Based Regression, Wrapper Approach Data Mining, Resampling Methods

Resumo

A extração de conhecimentos tradicional foca-se em encontrar padrões que são observados com mais frequência nos dados. No entanto, há problemas em que instâncias mais raras podem ser traduzidas em alta utilidade, como deteção de fraude e de catástrofes ambientais. Em cenários do mundo real, a tomada de decisão é frequentemente influenciada por essas regiões do espaço de dados, que não são muito populadas.

Os algoritmos de aprendizagem automática, geralmente focam-se em medidas comumente usadas, como o raiz do erro quadrático médio, o erro quadrático médio e o erro absoluto médio (para problemas de regressão). Estas medidas levam a que os modelos provavelmente obtenham grandes erros em instâncias mais raras. Portanto, as medidas são insuficientes quando a alta precisão é particularmente importante em áreas específicas do conjunto de dados, como aquelas com menor densidade. *Utility-based data mining* pressupõe custos não uniformes nos dados. Assim, estas abordagens poderiam ser usadas nos problemas mencionados anteriormente, desenvolvendo funções de utilidade que dão maior importância a instâncias raras, em vez das mais comuns.

A maior parte da pesquisa em *utility-based regression* foca-se em problemas de classificação. Desta forma, o nosso foco é em problemas de regressão. Nesta dissertação, propomos uma abordagem de *utility-based regression* que pode ser usada em problemas cujo o custo não seja uniforme. A ideia é alterar iterativamente o peso do conjunto de treino de acordo com uma medida de utilidade. Como esta abordagem não depende do domínio, pode ser usada para problemas de regressão e classificação.

Para testar o método proposto, usamos conjuntos de dados artificiais e UCI. Os resultados mostram que em 100% dos testes realizados, foi possível obter uma utilidade maior quando comparado ao algoritmo de extração de conhecimento usando o conjunto de treino original. Também desenvolvemos uma medida de utilidade específica ao domínio do caso de estudo no retalho para poder testar a abordagem proposta. O objetivo, neste caso, era prever vendas líquidas diárias usando a métrica criada. Obtivemos uma pontuação mais alta de utilidade em todas as lojas analisadas.

Acknowledgements

Firstly, I would like to thank my supervisor, Carlos Soares, for his guidance throughout this dissertation and for introducing me to the area of utility-based data mining. I am very thankful for all the countless questions answered and the provided scientific expertise that made this thesis possible.

I would also like to thank João Guichard and Miguel Arantes for sharing their vast knowledge regarding the retail industry.

I must also express my gratitude to Yassine Bhougassi and André Correia for supporting my work and being present when I needed them.

A sincere thank you to all my friends, colleagues and teachers who supported me and directly or indirectly contributed to my academic path.

Lastly, I send my special thanks to my family. Especially my mother, father, and brother for always believing and supporting me throughout my life.

This work is financed by INOVRETAIL, through the SONAE IM.Lab@FEUP - Research and Development Laboratory, within the project “2018/INOVRETAIL/Utility Regression".

Francisco Cagigal

"Information is the oil of the 21st century,"
"and analytics is the combustion engine"

Peter Sondergaard

Contents

List of Figures

List of Tables

Abbreviations

DT	Decision Tree
GA	Genetic algorithm
ML	Machine Learning
RF	Random Forest
RMSE	Root mean squared error
SVM	Support Vector Machine
UBDM	Utility-based data mining

Chapter 1

Introduction

Data-driven decisions are an essential way to achieve competitive advantage [23]. This is the process of making a decision which its basis lies within data analysis. One of the most predominant types of analytics is predictive analysis. It takes into account data previously recorded to make predictions of what might happen in the future. It is used in different areas such as marketing, financial services, communications industry, retail, pharmaceuticals, social networking, among others.

Usually, in real-world situations, the problem at hand may not require the discovery of frequent patterns spread throughout the data domain. Sometimes a more insightful data knowledge can be derived if we focus on specific parts of a data set. This knowledge can be translated into high utility as it is significantly impactful on decision making. Therefore, the business user has reasons to be more interested in a lower error on these specific parts of the data set domain rather than others.

One integral part of predictive analysis is model evaluation. It allows the understanding of how a model fits a set of data so we can compare different models. In order to evaluate models, there are standard measures such as Root Mean Squared Error (RMSE), accuracy, among others. Most of these measures do not take into consideration the different error costs throughout the domain of the target variable. Thus these measures are unfit for cost-sensitive tasks. Usually, Machine Learning (ML) algorithms use these metrics to optimize their predictions. Consequently, these algorithms fall short when the cost is not uniform across the domain of data.

Utility-based data mining is a type of cost-sensitive learning where the costs and benefits of the accuracy/error for each prediction are considered differently. This means that instead of attributing a uniform relevance to each instance in our data, such importance may vary with each instance. This data mining branch has a wide range of applications in areas such as fraud detection, catastrophes prediction, network attacks, rare disease diagnosis, among others [24]. On all these scenarios, there is a clear interest in a subset of its domain. Usually, the accuracy on rare examples is the most interesting from the user's point of view.

The proponent of this thesis is INOVRETAIL [1]. This is a company that provides data driven

[1] https://www.inovretail.com/

solutions to retailers. One of their services is predicting targets for stores. Targets represent a sales goal that an employee has to achieve during his shift. If the employee can meet the proposed goal, he is rewarded. The global target is the sum of all employees target during a day in a store. The proposed problem by INOVRETAIL, and the one which is addressed in this thesis, is to make predictions for net sales values in stores that take into account a store's potential to grow their sales. For instance, there are days where it is usual to have some operational complications. If these complications were to be properly addressed and solved, it would be possible to increase net sales. As such, an overprediction in this type of days does not hold the same significance as an underprediction.

1.1 Motivation

We are addressing the proposed problem by INOVRETAIL of net sales prediction. In this task, the error costs of predictions varies throughout data's domain. So, utility-based data mining should be regarded. However, most of these approaches focus on classification problems [27, 13, 12, 6, 31, 26, 4, 3, 10]. On the other hand, considering regression problems, research done on the topic is still on the early stages [1, 24]. Therefore, this thesis centers exclusively on utility-based regression.

1.2 Goals and contributions

The main goal of this thesis is to develop a wrapper based approach for utility-based data mining. This approach is algorithm-independent, which means it can be used with any ML algorithm.

Regarding our case study, the goal is to use this approach to address the net sales prediction's problem. In this task, the cost of underprediction and overprediction varies across the data's domain.As such, we need to define a function that is able to translate these misclassification costs. So, another goal of this thesis is the development of an utility measure for the retail case study.

1.3 Document Structure

The remainder of this document is structured as follows:

- Chapter 2, *Background*, starts by explaining the concept of genetic algorithms and detailing some strategies used on genetic operators. Finally, it introduces the concept of utility-based data mining for classification and regression problems;
- Chapter 3, *Approach*, starts by detailing the developed approaches and justifies some decisions made. Some experiments on artificial data used to understand the behaviour of the proposed method are also discussed;

- Chapter 4, *Empirical study*, starts by detailing the experiments done in UCI data sets and discussed the corresponding results. Then, it presents the approach followed on the case study to predict sales goals. Finally, the corresponding results are discussed.
- Chapter 5, *Conclusions & Future Work*, holds the main conclusions and points out possible future works.

Chapter 2

Background

This chapter presents the two main areas of interest for this project, Genetic Algorithms and Utility-based Data Mining.

2.1 Genetic algorithm

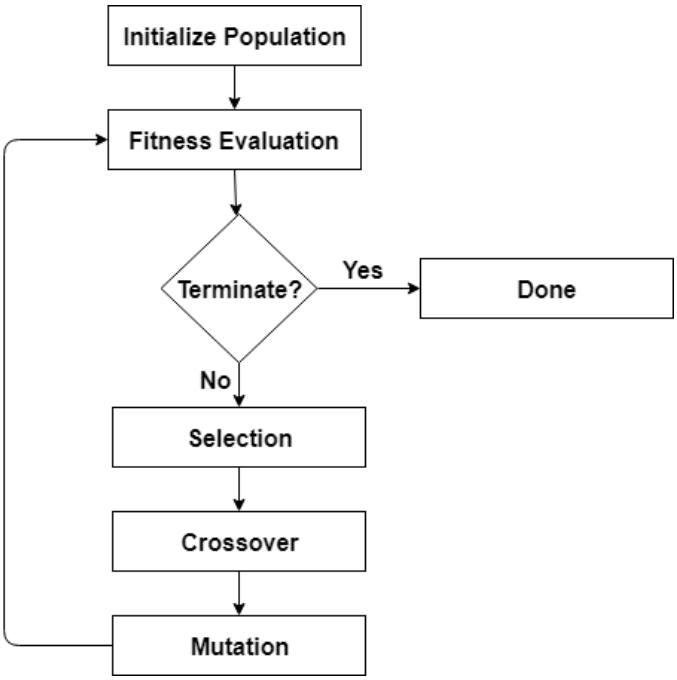
Genetic algorithms were introduced by John Holland in the mid-1960 [16]. Darwin's theory of evolution inspires these algorithms. They are based on the application of different evolutionary operators such as selection, crossover, and mutation to a population of solutions. These types of algorithms are often used in optimization and search problems. A common architecture of a genetic algorithms is presented in Figure 2.1.

Genetic algorithms try to combine exploration and exploitation approaches. Exploration consists of searching in a global area of the optimization function with the single goal of finding reasonable solutions that are still unrefined. This mechanism prevents the algorithm from being stuck in a local optima. On the other hand, exploitation entails the refinement of a solution that is assumed to be close to the global optimum. Of course, if exploitation is applied to solutions that are not near the optimal solution, it may be a risk that it can be stuck at a local optimum [20].

2.1.1 Fitness function

In every GA, there is a need to evaluate each chromosome according to the goal of the algorithm. This evaluation is performed by a fitness function. It is a domain-specific and user-defined function. This is an integral part of the algorithm as it allows to distinguish chromosomes that are good solutions from those that aren't. Naturally, chromosomes that carry food genes have a likelier chance of surviving throughout the iterations of the algorithm. Thus, there will be more and more fittest chromosomes as the number of iterations increase. As such, optimizing a fitness function will bring better solutions for the task, which is the goal.

Figure 2.1: Common architecture of a genetic algorithm.



2.1.2 Chromosome representation

In order to use a GA, the design of the chromosome needs to be specified. A chromosome is a set of genes. These genes carry the information of the solution. They translate an encoding method to a domain solution.

Usually, the representation of chromosomes is domain-specific. One of the most common ways of approaching this problem is using binary encoding. In this type of encoding, each gene corresponds to the value 0 or 1. It is also possible to encode chromosomes using a larger range of numeric values. Another type of encoding is string encoding, in which each gene corresponds to a string code.

2.1.3 Population generation

The population is an essential component of genetic algorithms. Chromosomes, which are also referred to as individuals are what composes a population. The algorithm changes values, with the goal of generating better solutions as the iterations go by.

Regarding its generation, there are two main discussion topics: the size of the population and how this population is created. Regarding the size of the population, some studies suggest a low population size can lead to suboptimal solutions [15, 22]. However, having a higher population size is computationally costly, and the algorithm may take more time than what is necessary [19, 18]. There are also some studies suggesting that the population size should be proportional to the complexity of the space search [32].

With regards to how the initial population is created, there are two main techniques: randomly generating genes or using a heuristic approach. Usually, random values are used to generate the population. However, it is also possible to use heuristic functions that may generate chromosomes with good fitness. These heuristic functions are associated with increases in maximum fitness when compared to random generation [28]. However, creating all the individuals of the population with a heuristic function can lead to premature convergence, in which turn leads to suboptimal solutions.

2.1.4 Selection

Selection is the process of choosing chromosomes to go to the next generation and discarding the remaining. This is usually a fitness-based process, in which the fittest individuals have a higher chance of being selected. The fitness function is domain-dependent and represents the translation of the problem.

Selection may include elitism. This is when the best individuals of the population are allocated in the next generation unchanged. As such, they do not undergo the process of mutation nor crossover. Saving these individuals for the next generation avoids the loss of the best solutions [9].

2.1.5 Genetic operators

The two most common genetic operators are crossover and mutation. However, it is not required that both of them should be used in a genetic algorithm.

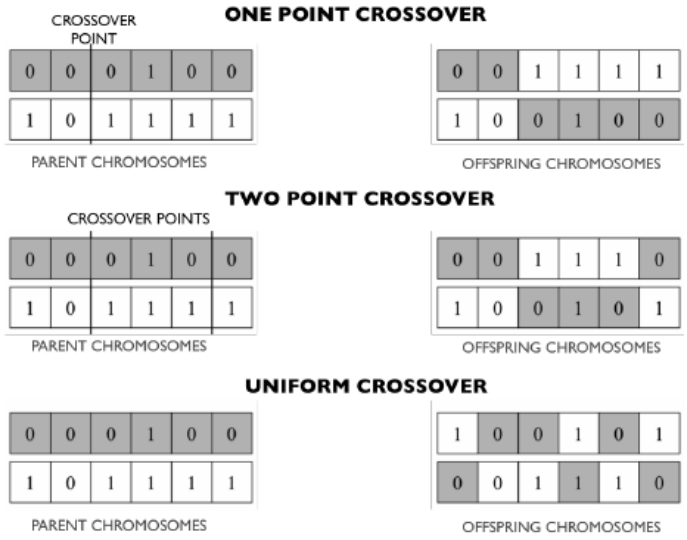
Crossover is the process of selecting two individuals to recombine their genes in order to create two offspring. Two standard ways to recombine genes is uniform crossover and k-point crossover and are represented in Figure 2.2. The latter happens when both parents are separated in the same k genes, and each child will have a genetic segment from one parent, followed by the next segment from another parent. In uniform crossover, each gene has an independent probability of being chosen from the corresponding gene of each of their parents.

Usually, mutation is a genetic operator used to maintain population diversity. It alters 1 or more genes in a chromosome. There are two standard approaches to mutation: intergene and intragene. In intergenic mutation, every gene has an independent chance of being mutated, while in intragenic mutation, the chromosome must first be randomly selected for mutation and only then can a gene of that chromosome be randomly replaced. Usually, low mutation rates can lead to premature convergence because of low genome diversity.

2.2 Utility-Based Data Mining

This section defines utility-based data mining and the research done in this field, including both regression and classification problems.

Figure 2.2: Example of crossover strategies reproduced from [21]



2.2.1 Definition

Conventional methods of measuring error such as MAE, MSE, RMSE are inefficient in real-world scenarios [24]. Measuring accuracy in imbalanced data sets can lead to a wrong perception of how good a model can be. This happens because having lower errors in common regions than rare regions of the data set may not be relevant in real-world problems [5]. Moreover, the cost of errors may differ between classes since the main objective could be to have a model that is good at detecting a type of class or outliers. For these scenarios, we need to use a different kind of error measurements that depends on the importance derived from the primary goal of the model. This kind of problems can be expressed as an utility-based data mining problem. Consider the example of a bank deciding whether or not to grant a loan to a set of clients. The goals of the bank can be to minimize the bank loss on granted loans. A bank loses money on a loan if the client is unable to pay for it. So an approach to solving this problem would be to maximize the accuracy of the class of clients that default a loan. Minimizing the error on this class allows the bank to lose less money with this class of clients in detriment of a higher error on the opposing class. Achieving a good accuracy on the former class can be done by having a higher error cost on that class than on the latter class.

Elkan [11] defines utility as a combination of benefits and costs that are specific to the domain of the application. There are two main methods for these type of problems: resampling the training sample and cost-sensitive ML. The first one corresponds to changing the weights of the examples in the training set before applying the ML algorithm. The latter is based on adapting already existent ML algorithms into being cost-sensitive or developing a new one that can maximize utility.

For regression problems, utility is defined as a function of both the error of the prediction and the relevance attributed to the domain of the target variable and of the prediction.

State of the art for this field is divided into regression and classification problems as follows in Section 2.2.2 and Section 2.2.3, respectively.

2.2.2 Classification problems

Regarding classification problems, there is a lot of study in different topics. This research entails relational costs in training set examples [27], quantification tasks [13], using evaluation measures as an utility function [12], imbalanced data sets [6], data collection [31], multi and binary classification tasks [3, 10].

In this Section, we will discuss Metacost in greater detail because it is a wrapper approach for classification problems.

Introduced by Pedro Domingos [8], Metacost is a wrapper algorithm that deals with different costs per class. The Bayes optimal prediction is the basis of this method which goal is to minimize the conditional risk $R(i|x)$, i.e., expected cost. The formula is as follows:

$$R(i|x) = \sum_j P(j|x)C(i,j) \tag{2.1}$$

where $P(j|x)$ is the probability that given an example x it belongs to class j and $cost(i,j)$ is the cost associated with the miss-classification of an example that belongs to class i predicted to belong to class j. This implies a partition of the example space into j regions, with class j being the least costly prediction in the j region. If the miss-classification of class j becomes more expensive than the miss-classification of other classes, then "parts of the previously non-class i regions shall be re-assigned as region i since it is now the minimum expected cost prediction" [29].

This method is a variant of the bagging method (also called bootstrap aggregating). Bagging is an ensemble algorithm that works as follow: given a training set of size s, it generates x new training sets of size s using replacement by sampling uniformly from the training set. Models are produced using each of the training samples created. The results are computed by averaging all the established models for regression or by voting for classification. The difference with Metacost lies on the fact that the number of examples of each generated training set may be smaller than the training set size [8]. It estimates all the class probabilities taking into account all the models made and relabels each training example with a rated optimal class using equation 2.2. Finally, the process repeats itself once again with the relabelled training sets to generate the final model.

$$class_x = argmin_i \sum_j P(j|x)C(i,j) \tag{2.2}$$

One advantage of this approach is that it is a wrapper method as it can be used with any classification learning ML algorithm without prior knowledge of it. The main disadvantage is the processing time since it has to run the ML algorithm s amount of times. Results demonstrate that this approach has reduced costs when compared to error-based classification and stratification. It was also concluded that it scales well with large data sets [8].

2.2.3 Regression algorithms

Despite extensive research on utility-based classification approaches, there is still a lack of research in regression problems. Utility is not often considered for these latter problems, even though its importance as it is common to have asymmetric costs [7].

In order for a problem to be considered to have an imbalanced domain and asymmetric domain relevance, one of the following properties must be verified [2]:

1. $L(y,y) = L(x,x) \nRightarrow U(y,y) = U(x,x)$: perfect predictions may output a different utility value;

2. $L(y_1,y_2) = L(x_1,x_2) \nRightarrow U(y_1,y_2) = U(x_1,x_2)$: the same error may output a different utility value;

3. A combination of the above options.

Ribeiro adressed the problem of utility based regression based on the concepts of relevance functions and utility surfaces [24].

2.2.3.1 Relevance functions

A relevance function is used in attributing importance to values in the data domain according to the application domain. A function $\phi()$ maps the target variable's values to the range $[0,1]$ where 0 corresponds to the minimum importance and 1 to the maximum relevance.

Branco et al. describes the benefits of this method as follows [2]:

- It is easier to define a relevance function based on the target variable's domain (y) rather than taking into account two variables $\hat{y}$ and y;

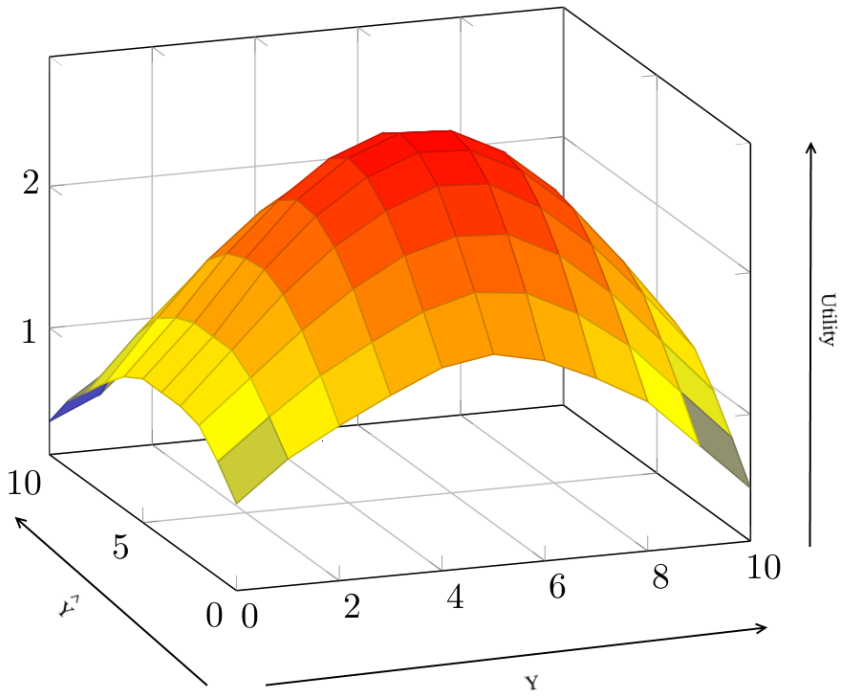
- Utility functions require more information to be defined when compared to relevance functions. The former needs a deeper domain knowledge to be specified.

2.2.3.2 Utility surface

Utility surface is a function that takes into account the relevance of the predicted values and the real ones as well as the error of the prediction and turns it into utility. In [24], utility is a measure of the error between the estimated and actual results as well as the relevance that the domain user attributes to $\hat{y}$ and y. Figure 2.3 illustrates an example of a utility surface.

This utility surface is what distinguishes a standard regression problem from an utility-based one. The former assumes the error ($L(y,\hat{y})$) is uniform across the data domain. On the other hand, the latter, besides depending on the same loss function, also takes into account the relevance of the variables $\hat{y}$ and y.

Figure 2.3: Example of an utility surface. reproduced from [25]



The conditional density of the target variable is used to estimate the utility of a prediction. This conditional density function can be seen in equation 2.3 where $f_{X,Y}(x,y)$ represents the join density function of X and Y, and $fX(x)$ is the marginal density of X [1].

$$f_{Y|X}(y|X=x) = \frac{f_{X,Y}(x,y)}{f_X(x)} \tag{2.3}$$

It was also introduced in [24] a utility measure that best suits utility-based regression problems as demonstrated. The formula is called mean utility and goes as follows:

$$MU = \frac{1}{n}\sum_{i=1}^{n} U_{\phi}^{p}(\hat{y}_i, y_i) \tag{2.4}$$

where n is the number of predictions generated, ϕ the relevance function, p the penalizing costs factor, $\hat{y}$ represents the predicted value and y the real observed value.

2.2.3.3 ubaRules

Proposed by Rita Ribeiro [24], ubaRules is an ensemble method applied to regression problems to optimize a utility function. Its main goal is to obtain an accurate model according to a specified utility measure. The author's intent was that it should be interpretable for the domain user because relevant values are usually associated with costly/beneficial decisions [24]. Thus the choice of opting for a rule-based formalism for the representation of the models as they are easily understandable. This ensemble method is divided into the following two steps.

Firstly, different regression trees are generated using various samples from the training set. Regression rules are extracted from these models. A regression rule is defined as follows [24]:

$$r_k(x) = U_k \prod_{j=1}^{\rho} I(x_j \in \chi_j^k) \tag{2.5}$$

In equation 2.5, χ_j^k represents a subset of the domain values for input variable j, x_j is the value of x on input variable j, I represents a function which returns 1 or 0 depending on if the argument inside it, is true or false, respectively. ρ is the number of input variables and U_k represents the constant prediction of rule k for the target variable Y. The rule is equal to U_k if every input variable satisfy the conjunction of tests proposed by the rule. Otherwise it is equal to 0.

Given that the rules have been obtained from different regression trees, the next step is to choose the rules with which to generate the final model. However the number of rules increases with the number of models used, and although it can improve the accuracy of the model, it may also reduce its interpretability as the complexity of the model increases with the number of rules. So there is a need for a process that reduces the number of rules by selecting only the best ones. Each rule is evaluated individually using a utility metric U. The rules are ordered in decreasing order of their estimated utility to form a ranking system. Finally, the best n rules are selected to generate the final ensemble.

2.2.3.4 Sampling approaches

Most ML algorithms focus on the most common cases on the data sets but may not cover regions that are of the most importance to the user.

In [30], Torgo et al. propose two approaches that are adapted from existing sampling methods. The first one consists of undersampling common values. The goal of this approach is to have a better ratio between common examples, which have reduced importance, and rare cases where its importance is significant.

The second one is an adaptation of SMOTE sampling. SMOTE is a sampling method used for unbalanced data sets in classification problems. It entails adding synthetic examples to the minority class so it has the same occurrence as the majority class. The authors propose that this method should be adapted to regression by defining regions of interest. This approach should oversample the region of interest, which functions as the minority class, and undersample the uninteresting region, which corresponds to the majority class.

Results show that these approaches are significantly better than using the original training set with the same ML algorithm.

Chapter 3

Approach

This chapter presents the proposed wrapper approach for utility-based data mining. It also details experiments performed on artificial data. Finally, it explains how some changes to the approach came to fruition.

3.1 Algorithm formulation

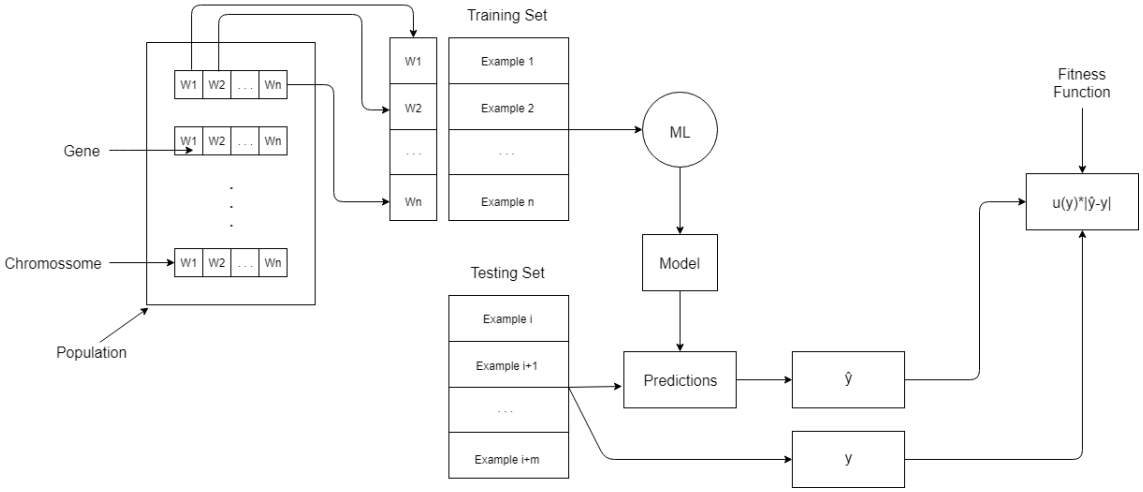
We are assuming that errors in a prediction are not always strongly correlated with its impact on the utility. As such, our approach consists of reweighting the examples in the training process in a way that more importance is given to examples with a higher impact on utility. We reweight the examples by resampling. The question is how how to define the right weights. We follow a wrapper approach, where the method iteratively searches for the best weights, evaluating solutions by learning and testing the model with the proposed weights. The search algorithm is a genetic algorithm.

A chromosome represents the weights assigned to each example (in the training data) and the fitness function is the utility of the model obtained with the training set reweighted according to the chromosome. Thus we could use a genetic algorithm for our resampling strategy.

3.1.1 Chromosome representation

To use genetic algorithms, we need to define what is the population, chromosomes, and genes in our problem (Section 2.1). Per the general definition, the population is a set of chromosomes (individuals). In our problem, a chromosome is used to transform the original training set in another that will be used for modelling. The chromosome has a fixed number of genes that is equal to the number of rows in the training set. Genes compose a chromosome and represent the weight of each instance in the training set. So each chromosome functions as a mask of the training set. For instance, if the first gene has the number 10 in it, the first instance of the original training set appears ten times in the new training set. In summary, each gene represents an example from the training set and its value (weight) corresponds to the number of times this example is going to be replicated in this set. Thus we would have the number of times each instance appears in the

Figure 3.1: Architecture of the proposed solution.



training set changing throughout the algorithm. The goal is to find the best combinations of genes that optimize a fitness function.

3.1.1.1 Fitness function

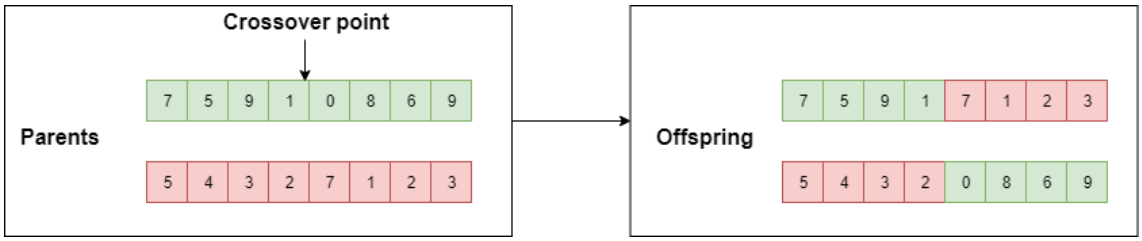
When we test the fitness of each chromosome, we use a fitness function. In this case, our fitness function is the utility evaluation metric. Firstly, to evaluate each individual of the population, we need to use the new generated training set to create a model. Then we measure the performance of this model on a testing set using a utility metric (3.1). The utility of a chromosome is given by equation 3.1, where i is an instance from the testing set, n the total number of cases in this set, y_i the real value of the target variable, $\hat{y}_i$ the prediction value, and $u(y)$ the importance attributed to this instance of a testing set. This formula represents the preference of the error in the domain of the target variable. Since this formula is proportional to the error on the predictions, we want to minimize its output. However, GA tries to maximize their fitness functions. To handle this issue, we used equation 3.2 to transform the minimization problem into a maximization one. At the end of the selection process, we have the same number of individuals that we had at the beginning.

$$U(c) = \sum_{i}^{n} u(y) * |(y_i - \hat{y}_i)| \tag{3.1}$$

$$US = \frac{1}{U(c)+1} \tag{3.2}$$

The following sections describe the GA operations such as selection, crossover, and mutation. We developed two approaches based on the same concept. The second was developed based on preliminary experiments with the first one but, for convenience, we describe both of them before the experiments. As such, for clarity purposes, the first approach is referred to as Approach I, while the latter approach is referred to as Approach II. These approaches differ in the selection

Figure 3.2: Example of 1-point crossover.



and mutation stages. Therefore, in the next sections, these operations are going to be divided into subsections for each of the strategies.

3.1.2 Selection process

This section details the selection process used for Approach and Approach II.

For Approach I, at the beginning of each iteration of the algorithm, the population must be evaluated according to a fitness function in order to use roulette wheel. The fittest individuals according to this function have a higher chance of being selected for the next phase. However, in our algorithm, we also guarantee the n fittest individuals are present in the crossover step. This process differs from the general elitist approaches as the chosen individuals go through crossover and mutation, unlike standard elitist methods. We chose this strategy as modeling of an individual is a costly process, and disregarding some individuals for crossover and mutation seemed wasteful.

As for Approach II, after evaluating the fitness of each individual of a population, these are ordered by their fitness value. The worst n individuals are removed from the population. The best n individuals are replicated once, creating two sets of these individuals. The first set goes directly into the next iteration without going through crossover or mutation processes. The second sample joins the rest of the population in crossover and mutation.

3.1.3 Crossover operator

In the crossover, we generate chromosomes by combining two of the parents of the population. Parents are randomly selected for the process. Some individuals may not participate in this task as they may not be selected. This operator is associated with a probability. Since each chromosome has an independent probability of being selected, that may not happen for all cases. There are many crossover strategies, but we chose to implement single point crossover (3.2). The reason behind this choice was the building block hypothesis [14]. This hypothesis consists of recombining building blocks, which are low order sequences of genes with above average fitness. In our case, the examples of the training set are ordered according to their target variable. As such, a building block could be a part of the domain which is given a high or low relevance.

3.1.4 Mutation operator

Mutation consists of replacing gene values by other values which are randomly generated. There are two types of mutation operators, intragene and intergene. Intragene mutation requires that an individual is randomly drawn for mutation and then one or more genes from that individual are changed to other randomly generated values. In intergenic mutation each gene has an independent probability of being mutated. In our algorithm, we opted for the intergenic mutation the mutations have a higher chance of being more equally dispersed between the chromosomes. Thus, we have a higher probability of changing more chromosomes per iteration.

3.1.4.1 Mutation process I

This process in Approach I is a common intergenic mutation with a fixed rate. Each gene has an independent probability of being changed to another value randomly generated.

3.1.4.2 Mutation process II

In this strategy, the mutation is not a fixed value but rather a varying one. At the beginning of the algorithm, the mutation rate is set to a high value. Then we must define a parameter i which represents the number of iterations needed for the mutation rate to drop while the best solution found remains unchanged. For instance, if i is equal to 10, it would require ten iterations for the mutation rate to drop. However, if the utility of the fittest individual increased, the counter is reset. In this situation, it would take another ten iterations for the mutation rate to decrease. The mutations rate is reduced by a determined factor r. It stops dropping when it hits the minimum mutation rate specified. Algorithm 1 translates the specified operator into pseudo-code.

3.2 Utility measures

Utility functions quantify the importance of different areas of the data domain. The utility function is typically determined by the application domain. However, to validate the approach we would like to test it on artificial data and also benchmark data sets. This is also important for reproducibility, as the data from the case study used in this project is not public domain. Thus, we need to arbitrate a utility function for the experiments with artificial and benchmark data. As in some real-world problems, there is a more significant interest in extreme and rarer values on the data domain [24], we followed the same line of thought. As such, we give more importance to values that are on the extreme of the data domain rather than the middle of it. In our methodology, we are relaxing the definition of a utility function to a relevance function in order to simplify our experiments.

We test this approach on data sets which target variable has more examples in the middle of its domain rather than the extremes. For instance, if the domain of the target variable resembles a normal distribution, it is guaranteed that we have more examples in the middle region and fewer instances in the extremes. Thus it allows us to accomplish our goal of attributing higher importance

Algorithm 1: Mutation for Approach II.

```
mut_i ⟵ initialmutationmutationvalue;
mut_f ⟵ lowestmutationvalue;
dec_r ⟵ decayingrateofthemutation;
n ⟵ numberofiterationswithoutimprovementstodecreasethemutationrate;
begin
    counter ⟵ 0;
    best_solution ⟵ 0;
    current_mutation ⟵ mut_i;
    while genetic_algorithm_termination_condition do
        if current_solution > best_solution then
            best_solution = current_solution;
            counter = 0;
        else
            counter += 1;
        end
        if counter == n then
            current_mutation = current_mutation x dec_r;
            counter = 0;
        end
        if current_mutation >= mut_f then
            current_mutation = mut_f;
        end
    end
end
```

to extreme values rather than values located in the middle of the data domain. If this does not occur, it will not bear meaning to use this utility measure to demonstrate how our algorithm works as the values on the extremes would be common and easily predicted. Moreover, the utility function should adapt to the scale of the target variable. As such, it is data-independent, i.e., can be used with any data sets without previous knowledge.

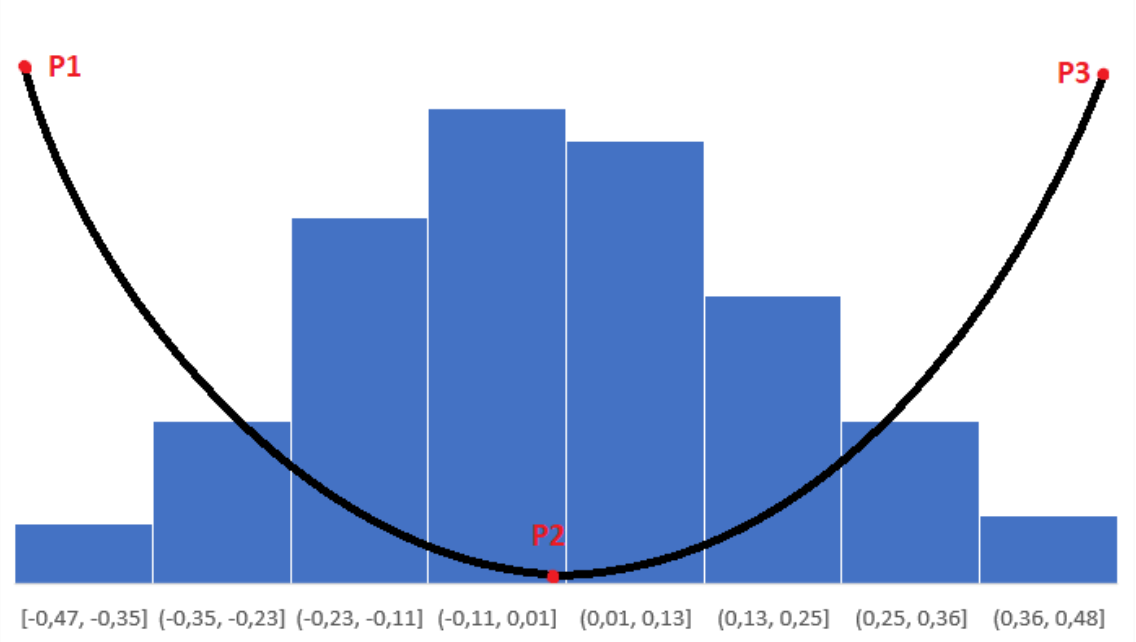
Figure 3.3 represent how the variable Y is distributed across the domain for an artificial data set. As previously said, it is essential that these data sets' distribution is similar to a normal distribution to create our utility measures. As we can see from that figure, the closer the examples of Y variable gets to the extremes, the less frequent they are. This allows us to create a utility metric by giving more importance to extreme cases and lesser significance to examples around the mean. Taking into account the distribution in figure 3.3, we can calculate the displayed utility function in three steps. Firstly, we find the minimum ($P1$), maximum ($P3$) and median ($P2$) value of the target variables (Y). Then we calculate a positive quadratic function, which we will call $F1$, using $P2$ as the minimum and $P1$ as an auxiliary point. Lastly, we calculate a quadratic function $F2$ with the values $P2$ and $P3$ following the same logic. We end up with two equations $F1$ and $F2$. When evaluating a model on a testing set, if the value of the target variable is lower than the median, we use $F1$, while if its higher we use $F2$.

The reasoning behind this decision of having two quadratic functions was to correct the possibility of our target variable being skewed towards one of the sides. If the right side of the median, which corresponds to the side which the maximum is in, represented considerably higher importance than the left side it would devalue the importance given to the left side and thus would not meet the purpose of the experiments. For instance, in the previously mentioned data set, if the maximum of the Y variable is 1, and we would only use one quadratic equation, it would result in a significant discrepancy between the two extremes of the data domain.

As we can notice, figure 3.3 does not have a vertical axis. Its absence happens because we want to be able to set y-coordinate of the aforementioned points to whatever we see fit. With that in mind, we established that $P2$ would still be given a positive value as its y-coordinate because we want it to carry importance higher than 0, however small that influence may be. Regarding the values of $P1$ and $P3$, we used different values as its y-coordinate. By attributing different values, we could be able to change the curve and analyze how it behaves with our approach. This parameter will be referred to as K. Equation 3.3 details the formulation of the utility function with b being the minimum importance given to the the median, k the maximum importance given to both extremes, and m the values of the maximum and the minimum of the domain depending on what quadratic function is being formulated.

$$y(x) = (((K-b)/(m-median)^2) * (x-median)^2 + b) \tag{3.3}$$

Figure 3.3: Frequency distribution graph of the target's variable of an artificial data set with a possible relevance function. P1 - Minimum value; P2 - Median; P3 - Maximum value;



3.3 Experiments

In this section, we will discuss some experiments done with artificial data and the obtained results. These experiments were made so that we can validate our approach. We are also analyzing the best training set obtained, i.e., the weights attributed to the examples in the training set.

3.3.1 Experiment I

Firstly we are going to test the approach using the artificial data set. Let us consider the construction of the following data set. Firstly, we randomly create a list of 1000 numbers that range from -0.5 and 0.5 of mean 0 and standard deviation of 0.2. We will refer to this list as Y. Then we create the X variable that results from cubing all the numbers in Y. The final data set consists of aggregating the variables$X$ and Y, where Y is the target variable. Thus Y should have fewer examples on the extremes and more in the middle. This data set is randomly divided into a testing set and training set. 75% of the examples are placed in the training set, and the remaining 25% is placed in the testing set. Figures 3.4 and 3.5 represent the obtained training set and testing set respectively.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor

gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed,

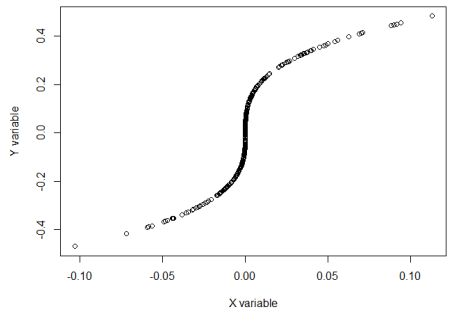


Figure 3.4: Training set for the artificial data set.

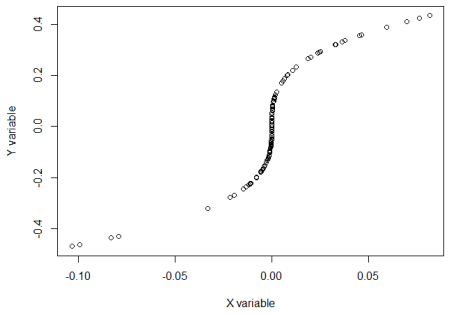


Figure 3.5: testing set for the artificial data set.

volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

For this experiment, the parameters used are detailed in Table 3.1. The initial population is generated randomly from values of 5 to 20, with a mean of 10 and a standard deviation of 3. All elements of the population are available for crossover and mutation. The ML algorithm used was linear regression. Regarding the utility metric, we used the one explained in Section 3.2 with a simple modification: all examples in the training set which Y variable is between -0.25 and 0.25 do not contribute to the utility value, i.e., their error is not considered for utility purposes.

Parameters for tests run	
Parameter	Test
Training set size	5
Population size	10
Elitist rate	10%
Crossover rate	75%
Mutation rate	0.2%
Number of iterations	10000
K	20

Table 3.1: Parameters used in experiment I.

Figure 3.6: A) Result of linear regression on the original training set. B) Result of linear regression on the training set obtained by our approach.

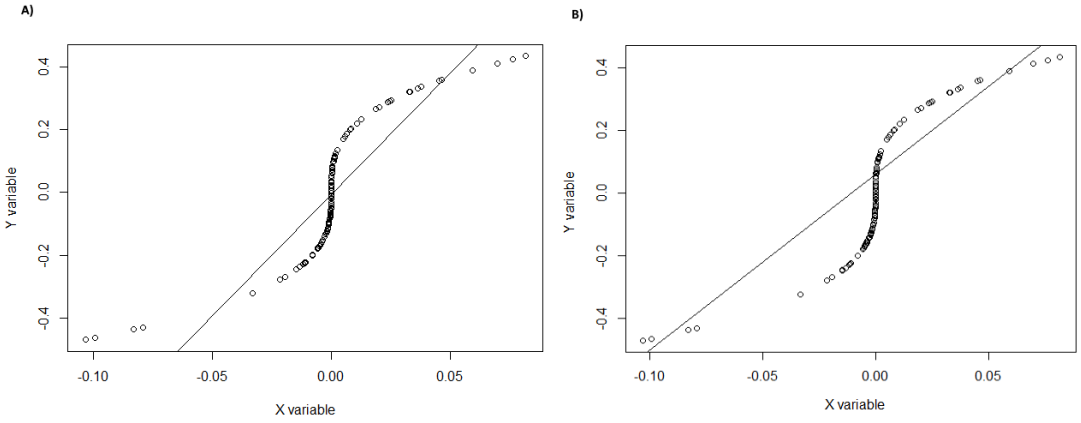
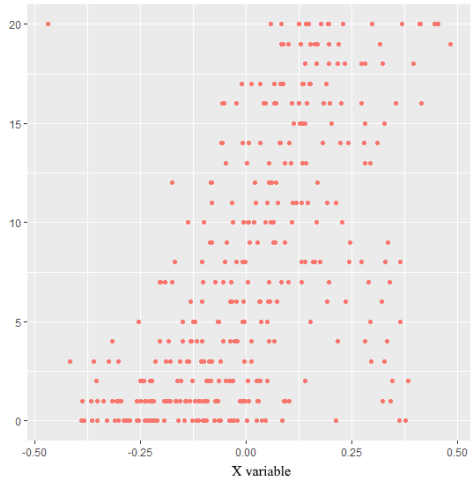


Figure 3.6 compares our approach I with a linear regression algorithm and the linear regression model with the original training set. As it was expected using our method, we got a model that better fits our utility function. This fitness can be verified by comparing the angle between the line and X axis wherein our approach the angle is smaller. Consequently, the line is closer to the extremes of the target variable (Y), which we attribute higher importance. Moreover, the value of utility using our approach is 0.049 while using the original data set is 0.026. Even though there is a small difference between utility values, when looking at the produced models, we can see a clear difference.

Figure 3.7 shows the best distribution of weights per training set example we were able to achieve. We expected that the weights would reflect our utility measure. So if an example was in an area of great importance, its weight should be higher. However, this does not necessarily happen (Figure 3.7). We found out that it indeed gives higher weights to the extreme examples, though in the left side of this figure we only have one instance with maximum weight and right next to it there is a significant drop. This fall can be explained by looking at our training set and testing set distribution in Figures 3.4 and 3.5. We can see that the training set has only 1 example near the highest value region while the next one is still at a relatively large distance from the former. We believe that if the algorithm had given higher importance to the second lowest example in the training set, according to Y variable, it would lower the utility value formerly obtained. This assumption was proven correct by manually changing the obtained distribution. Looking at the right side of the figure, we can see that there are a lot more examples that are near the maximum weight. Comparing with the left side, there are more examples in higher importance areas on the right side, where the Y value is higher than on the left. This could be used to justify the higher weights on that region. We can also verify that there is a drop in weights just like it happened on the left side of the figure. The most exciting aspect and the one we were not expecting is the region in the middle. There are a lot of examples that have a high weight associated with them

Figure 3.7: Weight attributed to each example in training set using our approach.



even though they are in the area that doesn't contribute to the utility function. This region is a little skewed to the right because there are more values on the right side than on the left in the training set. Our hypothesis for this result is that examples that are next to regions of really high relevance affect more the resulting model than examples on the middle part where the Y variable is close to 0. So the weights attributed to the middle would work as a sort of balancing mechanism to the linear regression model.

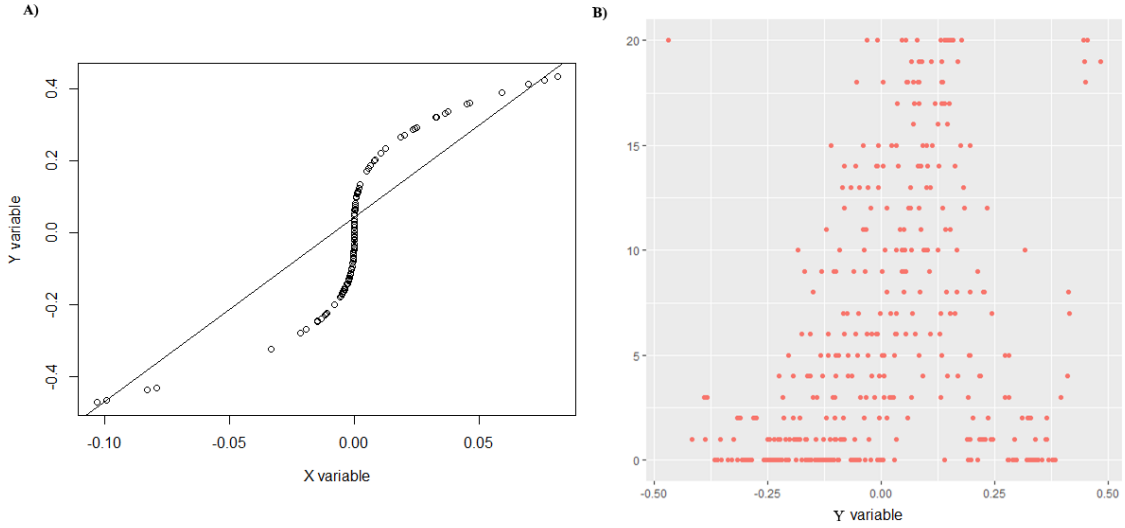
We decided to run a few more tests to see if this behavior was consistent. The tests are as follows:

- Test I - We used the same approach as before with the same training and testing sets. However in this case when calculating utility we would only look at two values from the testing set - the minimum and the maximum. The genetic operators used are the same as the first test. It would be expected that the regression model would come closer to both extreme points.

- Test II - We combined the former training and testing set as our new training set and created a new testing set. This sample is composed of 100 values which the Y variable ranges from -0.5 to 0.5 equally separated by 0.01 and the X variable is the cubing of the Y values. The genetic operators and utility function used are the same as the first test. As the test set is symmetric, it should be more noticeable the appearance of a pattern on the weight distribution.

The resulting model and distribution of weight for Test I are represented in Figure 3.8. We can observe that the line is close to the points that represent the maximum and the minimum of the Y variable. Thus it is working as would be expected. However, it is unlikely that this is the best model possible to be achieved. When looking at the obtained distribution of weights, we can see a clearer image of what was captured before. It is attributed a higher importance to both of

Figure 3.8: Results regarding Test I: A) Resulting model using our approach. B) Distribution of weights obtained in the training set.



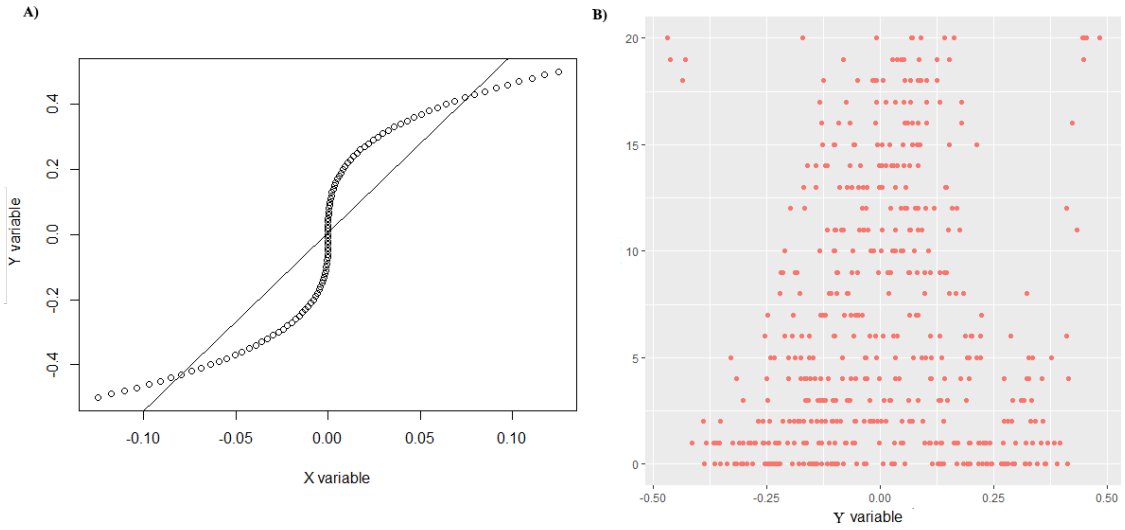
the extreme regions, right and left. Then appears a fall in relevance attributed to the next closest examples. After that, it starts increasing again as it gets nearer to the middle.

The resulting model and distribution of weight for Test II are represented in Figure 3.9. When analyzing the obtained model, we can see that the line is inclined towards the region of most significant importance.

Concluding this experiment, we can verify that Approach I behaves similarly throughout the different tests. So we hypothesize that the fall that happens in the distribution is due to those areas being the ones that most negatively affect utility. On the other hand, the ones in the middle does not hurt as much because they are near the region the line should cross. They can also work as a balancing mechanism that applies small changes to the linear model without affecting it significantly. Consequently, it makes sense that the algorithm is not prioritizing these changes and instead is focusing on the regions where the dip in weights happens.

Regarding the obtained models, we can notice the approach is working towards a good solution, but there is not clear evidence of how good the obtained model is when compared to the optimal solution. This uncertainty is due to the range of possible combinations available. Using a training set of 375 examples in which the weights may vary between 0 and 20 gives 21^{375} possible combination of possible training samples. Since this number is too high, it is computationally unfeasible to test all the possible models. However, it is our opinion that the algorithm is not being given enough time to compute to achieve a better solution. This inefficiency mainly happens because there are too many values that may need to change.

Figure 3.9: Results regarding Test II: A) Resulting model using our approach. B) Distribution of weights obtained in the training set.

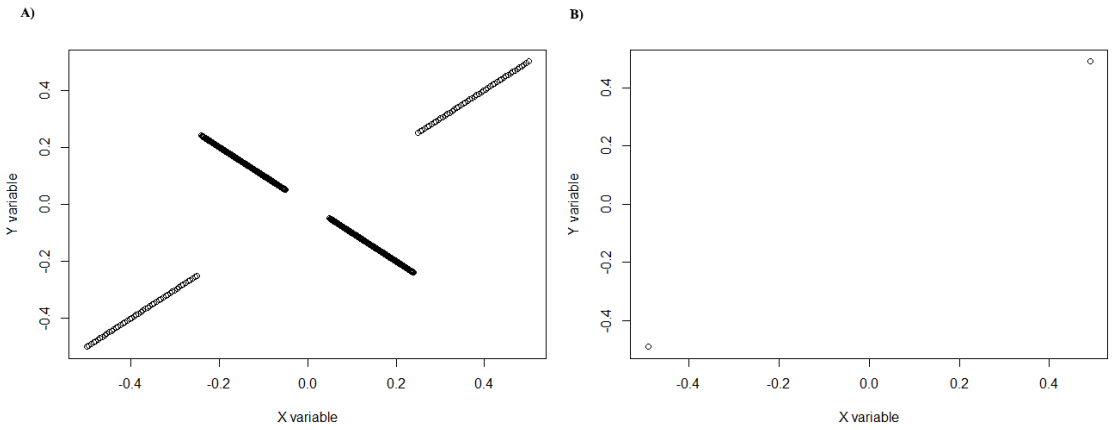


3.3.2 Experiment II

In the second experiment, we addressed some of the limitations of the previous one (section 3.3.1), namely regarding the difficulties in evaluation and the complexity of the search space.

Regarding the difficulty in evaluating of the model and how close it is to the best possible solution, we decided to create a new data set in which we would be able to know which is the perfect model. The data set is organized as follows: for the Y variable, we have four aggregated sequences. The first one is composed of 450 values that range from 0.05 to 0.24 equally distanced between themselves by 0.000423. The second sequence is constituted by 450 values that range from -0.05 to -0.24 equally distanced between themselves by 0.000423. For both of these sequences, the X variable corresponds to $-Y$. The third sequence consists of 50 values that range from 0.25 to 0.5 equally distanced between themselves by 0.0051. Finally, the last series comprises 50 values that range from -0.25 to -0.5 equally distanced between themselves by 0.0051. For both of these sequences, the X variable is equal to the Y variable. This data set will be used for training of the linear regression model. Regarding the testing sample, it is composed of only two examples. These examples are the points which the coordinates are (-0.49,-0.49) and (0.49,0.49). A representation of the training and test sets can be observed in Figure 3.10. When looking at the training set, we can see that the vast majority of examples (90%) forms a line with a negative slope. Thus these examples will have significant impact in a linear regression model. This impact can be verified by looking at the resulting model using this training set in Figure 3.11. However, when looking at our testing set, we can notice that the generated model has a very bad utility value because the line does not come close to any point on this set. From the perspective of the utility function defined, we expect that the weights in the middle would gradually decrease to 0 while

Figure 3.10: A) Data set that will be used for training. B) Data set that will be used for testing.



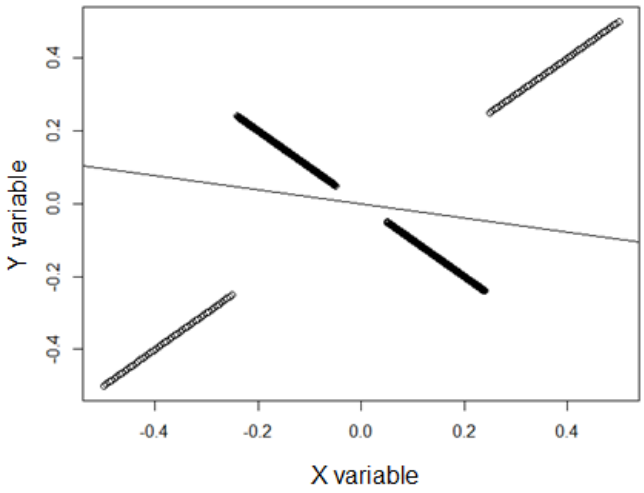
the weights in the extremes would increase significantly. Using this training set to learn a model that maximizes utility, the weights of examples with $Y \in [-0.24, 0.24]$ should be very low or even zero.

Concerning the search space, we carried out two tests. The first one (Test I) is a small scale of the training set mentioned earlier, containing only five examples where there are three in the middle region and one in each extreme. This test should be simple enough that the algorithm should find the best model within a few iterations. The second test (Test II) consists of changing a percentage of the initial population weights to zero. The population is randomly generated as detailed in Section 3.3.1. However, after we have obtained the initial set of individuals, we change a fixed percentage of values to zero. This is done in a way that guarantees that the set of weights for each example in the initial population include a zero and a non-zero value. In other words, for each training example, there will be at least one chromosome which has a zero value in the corresponding gene and at least one other chromosome with a non-zero value in the corresponding gene. What we are expecting is that by setting a large number of weights to zero it would decrease the computational effort of the algorithm to find the best solution. The parameters for both tests in this experiment are defined in Table 3.2.

After running Test I, we were able to obtain the results shown in Figure 3.12. As expected, we obtained the perfect fit for our data. The values in the middle region were all given weights of zero, while the two values on the extremes were given non-zero values. As a consequence, the resulting linear model crosses the two points. Thus the model is achieving the maximum utility. Since this test only had five examples in the training set, it only serves as a proof of concept.

For Test II we ran the algorithm with different percentages of zeroes per individual: 99%, 95%, 90%, 80%, 60% as is specified in Table 3.2. What we were expecting is that as we decrease the percentage of zeroes, it would become harder and would take longer for the algorithm to find the best distribution. Figures 3.13,3.14,A.1,A.2 and A.3 represent the results obtained from this Test II. With a 99% of zeroes per individual, we can obtain a very good model as Figure 3.13 shows.

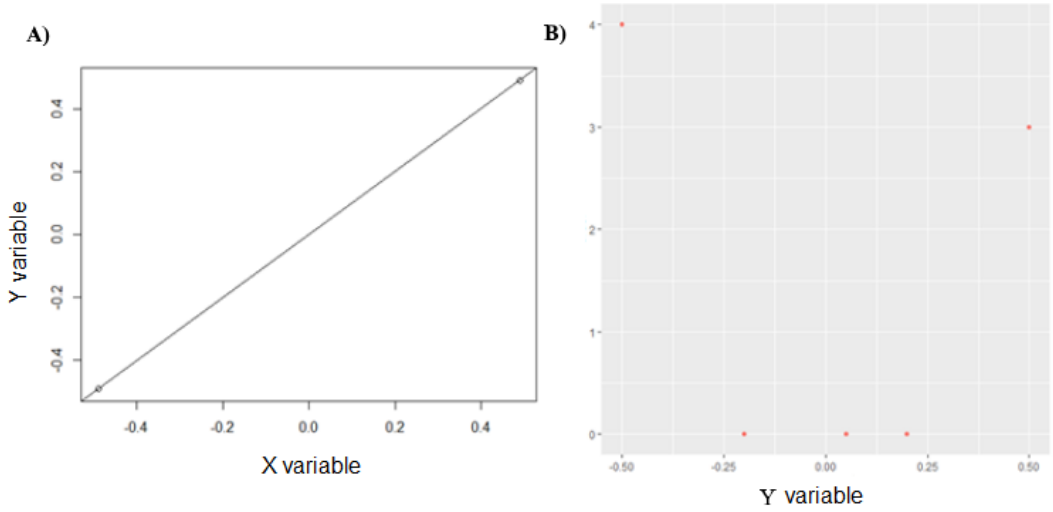
Figure 3.11: Linear regression model using the original created training set.



Parameters for tests run		
Parameter	Test I	Test II
Training set size	5	1000
Population size	10	100
Elitist rate	10%	10%
Crossover rate	75%	75%
Mutation rate	0.2%	0.2%
Number of iterations	10000	10000
K	20	20
Percentage of zeroes	non-applicable	99%,95%,90%,80%,60%

Table 3.2: Parameters that were used to run both tests.

Figure 3.12: Results regarding Test I: A) Resulting model using our approach. B) Distribution of weights obtained in the training set.



At first sight, it resembles a perfect model because the line appears to cross those two points in the testing set. But there still exists an error however small it may be. Thus this model is not perfect. This inference is corroborated by looking at the distribution of weights and notice that there are two examples in the middle that are given non-zero weights. Thus the utility obtained is close to the maximum value but has yet to reach the optimal solution.

Interestingly enough, when we try to modify the weight of each of those examples individually to zero, we obtain a lower value of utility. This occurrence means that the only way that we can achieve the perfect model is to change both of those weight to zero at the same iteration. This training set has 1000 examples in it, and with a mutation rate of 0.2%, we can expect an average of two mutations per individual and iteration. Moreover, these two mutations should occur to the individual that has this distribution. Also, the two examples of the training set must be those specific two examples that have a non-zero value. Furthermore, the mutations that can range in the interval $[0;20]$ must randomly select the value zero for both of those examples. We can see that although it is not impossible to achieve the best model, it seems rather unlikely that we would reach it in a reasonable time.

When comparing Figure 3.13 with Figure 3.14, we can see that the obtained linear regression model is pretty identical. Nonetheless, the latter is slightly worse, with more errors that generate a lower utility value. Furthermore, there are more examples in the middle region that are non-zero, which justifies the lower utility. One thing to notice is that almost all those non-zero weight values are near the very middle of the graph. The closer the Y variable is to 0.05 and -0.05 which are the absolute minimums possible obtainable from our training set, the less impact they have on the model. So the algorithm is "cleaning" the examples that are on the outer sides of the middle region first because they have more of a negative impact on the model.

Figure 3.13: Results regarding Test II with 99% of zeroes per individual: A) Resulting model using our approach. B) Distribution of weights obtained in the training set.

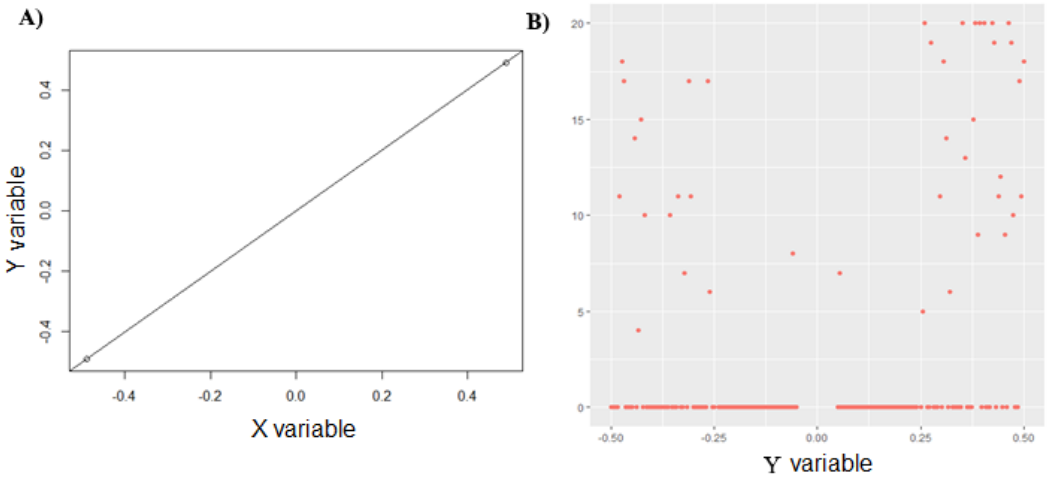
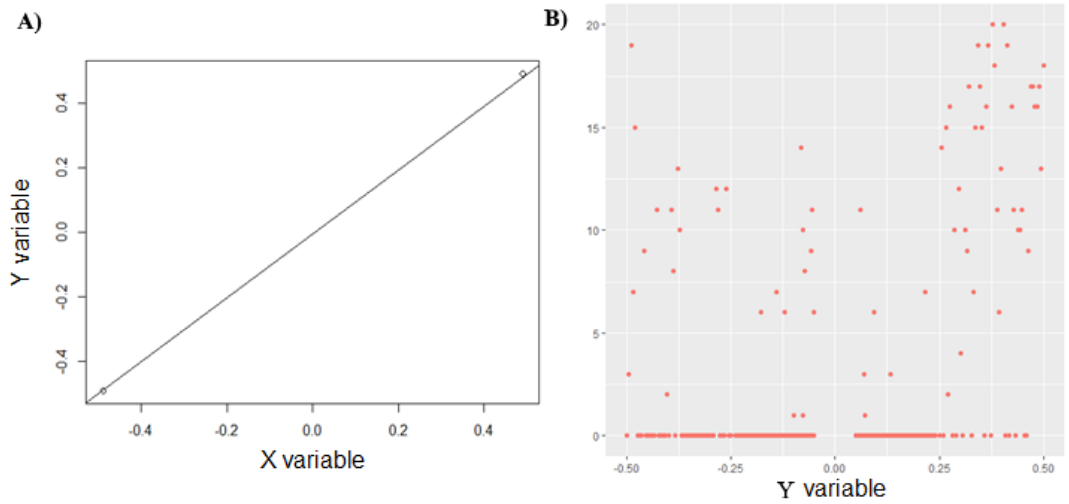


Figure 3.14: Results regarding Test II with 95% of zeroes per individual: A) Resulting model using our approach. B) Distribution of weights obtained in the training set.



Analyzing Figures A.1,A.2 and A.3, we can see that the quality of our model decreases when the percentage of zeroes given to the individuals of the population also decreases.

One of the main problems before this experiment was evaluating the resulting model. Using a simple testing set like it was demonstrated in this experiment eased the evaluation of the model because we know how the best model would look like and how its weight is distributed across the training set. The experiments also confirmed that the complexity of the search space affects the behaviour of the algorithm. They also showed that by increasing the number of zeroes in the initial population allowed the algorithm to work faster. Even though the best model was not achieved, there were some that came close to it.

However, another problem was found when analyzing the distribution of weights in this experiment. It resides in randomly choosing which example from the training set should change its weight. The algorithm assumes that the weights are independent. That is, when replacing the weights in the middle region to 0 one by one, it would increase the utility value. This assumption is not valid, as demonstrated by the analysis of Figure 3.13. We need to handle these dependencies in the training set (in this case, a sort of balance in the training set) so we could be able to obtain the fittest model to our testing set.

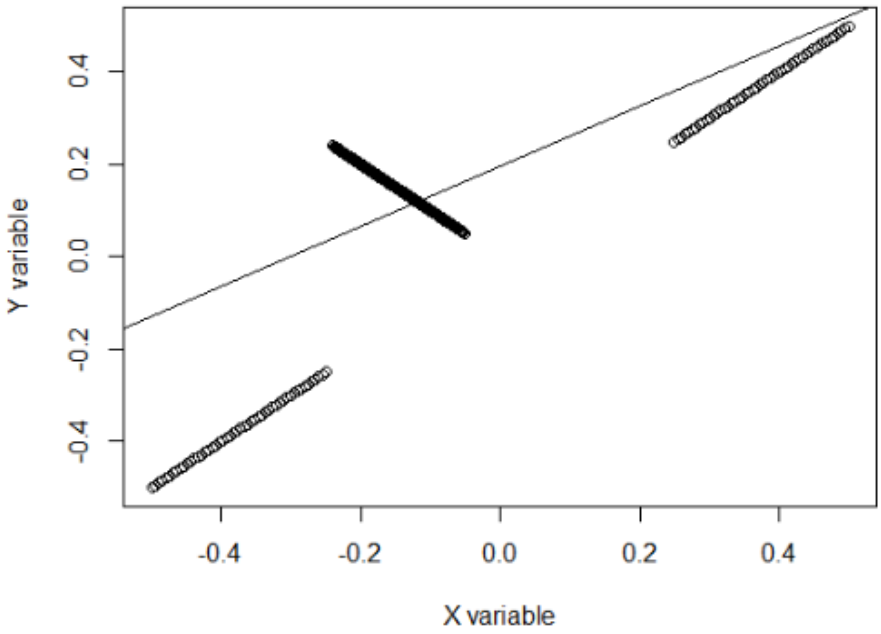
3.3.3 Experiment III

In the previous experiment detailed in Section 3.3.2, we observed that dependencies between the weights of examples may affect the behaviour of the proposed approach significantly. In the particular case of the artificial data we used, there seems to be a balance between the weights of the training examples that is important. The first analysis in Figure 3.13 reveals that those two examples in the middle region of the distribution of weights are one in each side of the area. This separation could mean that the reason we aren't able to increase utility by changing each of those examples' weights individually to 0 is due to a dependency between these two points that are located on opposite sides of the line $y = 0$. So we can hypothesize that if we remove one set of the middle region containing 450 examples we can also remove this dependency. Thus the algorithm may be able to randomly change the weights of the central area without decreasing the utility value.

The test set used in this experiment is the same as in Section 3.3.2 and is depicted in Figure 3.10. Regarding the training set, it can be obtained by removing all examples from the previous experiment which Y variable ranges from $[0,05;0,24]$. The resulting training sample which is used in this experiment as well as the resulting model is represented in Figure 3.15.

In this experiment, we will apply the same strategy as in the previous one regarding replacing the initial population with a percentage of zeroes. It is expected that the distribution of weights would suit better when compared to the previous experiment. This expected result is because there are fewer examples in the training set. Moreover, as we removed a part of the training set that could generate some balance, it is hypothesized that now the algorithm could lower the examples' weight from the middle region without any constraint. The parameters used to run these tests are shown in Table 3.3.

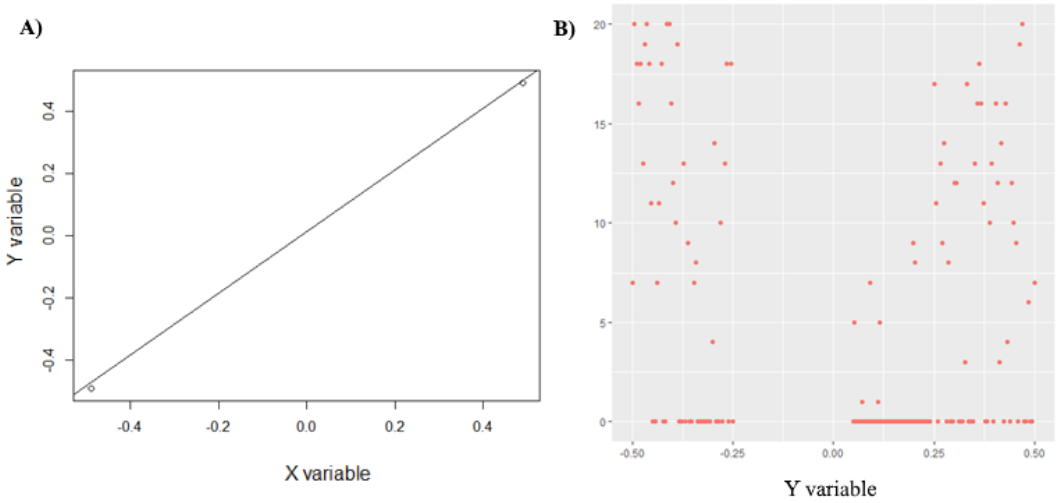
Figure 3.15: Training set used in experiment III and corresponding linear regression model.



Parameters for tests run	
Parameter	Test I
Training set size	550
Population size	10
Elitist rate	10%
Crossover rate	75%
Mutation rate	0.2%
Number of iterations	10000
K	20
Mutation values	[0;21]
Percentage of zeroes	90%,80%,60%

Table 3.3: Parameters used to run Test I of experiment III.

Figure 3.16: Results regarding Test I of experiment III with 90% of zeroes per individual: A) Resulting model using our approach. B) Distribution of weights obtained in the training set.



The results of this experiment is shown in Figures 3.16, B.1 and B.2. Comparing these results to the ones obtained in the previous experiment (Figures A.1, A.2 and A.3) we observe that there are fewer examples' weight that needed to be changed to zero in this experiment. This observation is more noticeable in the case of 90% replacement of the initial population by zeroes. It was part of the expected behavior since there are less 450 examples in the training set. Thus the number of values that need to be changed is reduced to half. However, it was expected that we would be able to achieve better models. It was also predicted that the number of non-zero weights in the middle region from the test with 80% of zeroes would be higher when compared to the 90% test. What was not expected was this steep increase in the number.

Since these results were not wholly expected, further analysis of the training set was needed. Manual manipulation of the weight distribution obtained from the test with 80% of zeroes was performed to understand why we were getting so many non-zeroes. After some analysis, it became apparent that it was not guaranteed that changing the weight of examples in the middle region to zero would increase utility. Thus the dependency in the training set that was intended to be eliminated is still present. However, it is present in a different way than before. In this case, we can change individually every weight to zero and guarantee an increase in utility if a determined order of replacement is followed. This order represents the sequence in which each individual weight is changed to zero. It does not mean that there is only one possible way of replacing the weights but rather a determined value for changing. For instance, if we change the weight of an example that is lower than the determined value, we will decrease utility. On the other hand, if we change the weight of an instance that is higher than the specified value, we will increase utility. So the hypothesis that the dependencies in the set of weights would be removed if we changed the sample is not entirely truthful. We were indeed able to remove the previous "balance", but we uncovered another one.

Parameters for tests run	
Parameter	Test I
Training set size	550
Population size	10
Elitist rate	10%
Crossover rate	75%
Mutation rate	0.2%
Number of iterations	1000
K	20
Mutation values	[0;1]

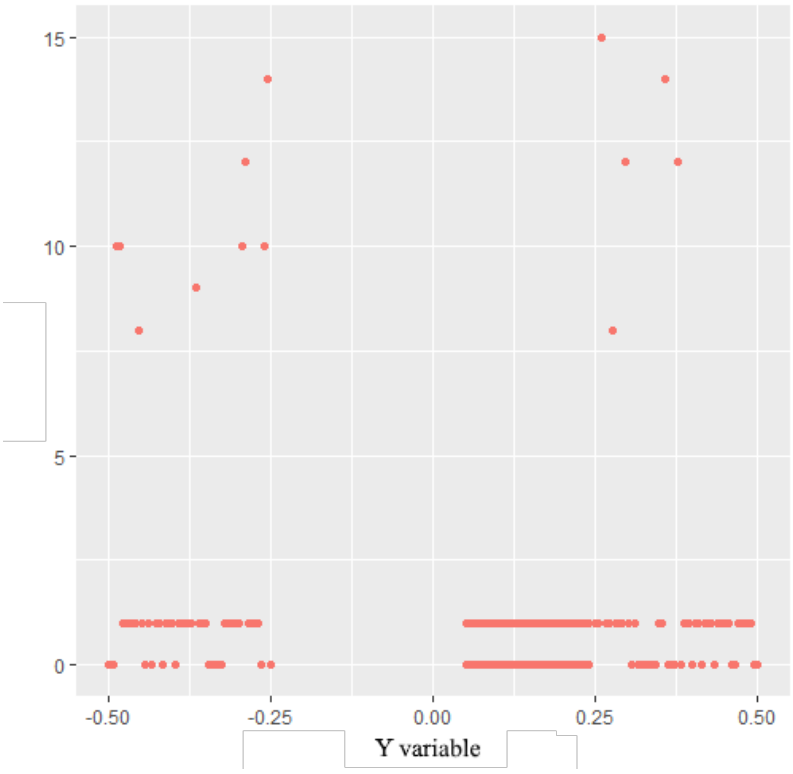
Table 3.4: Parameters used to run Test II of experiment III.

Since the goal of this experiment was to reduce the impact of dependencies in the set of weights by manipulating the training sample, the test was not a success. Another way we could increase the chances to achieve our goal is to decrease the range of possible values in the mutation. In our previous experiments, these values' range is $[0;20]$, but we know that in this experiment we can get the perfect model with mutation values ranging from 0 to 1. So when it occurs a mutation, it would only have two possible values to change instead of twenty-one. Thus it would increase our chances of finding the best model. This idea is the basis for Test II of this experiment. The same training set and testing set are used, and the parameters are the same as Test I, which are displayed in Table 3.4. However, in this test, the replacement of the initial population by a percentage of zeroes is not applied.

The results for Test II are presented in Figure 3.17. It can be verified that all the weights in the middle region are either zero or one. On the other hand, the extreme areas have some examples' weight higher than one, which confirms that these examples were not mutated. Since 82% of the training set examples are in the middle region and 9% in each extreme, it would be rather unlikely that every instance in the central area was mutated without some utility measure that resembles the one in use. Thus we can conclude that the algorithm is trying to achieve its goal of a perfect model in attempting to mutate each gene that damages the algorithm to zero. However, we still aren't able to achieve our goal. There are still a reasonably high number of examples in the middle region, which weight is higher than zero.

When analyzing our selection approach, we guarantee that the fittest 10% individuals of the population are selected while the remaining 90% are randomly selected according to a fitness function. And as usual, the fitness function gives a higher chance of selection to the fittest individuals. At the end of the selection process, all the individuals are set to go to the crossover phase and afterwards to the mutation phase. In these two phases, the individuals may be changed. In the next generation, we can have a very different set of individuals from before. One of the individuals that may change is the fittest one. This individual may change its utility for better or worse. Therefore using this selection approach, it is not guaranteed that the next population will have an increase in efficiency. When the next generation's utility decreases, we lose the fittest chromosome which may be hard to recover. If there are successive iterations where the fitness is decreased, then it is

Figure 3.17: Distribution of weights obtained for the training set in Test II of experiment III.



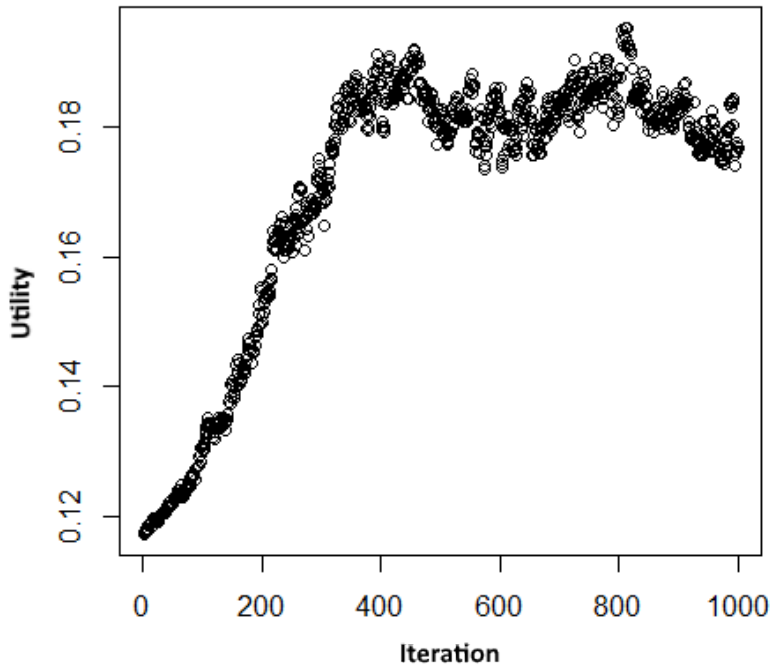
harder to regain the distribution of the fittest individual from generations before. The reason being that if this individual was the fittest one and it loses some utility, it may stop being the fittest one in the next generation. And if this happens, this individual may even fail to be selected for the next iteration. So what may be occurring is that throughout the algorithm iterations, the utility of the fittest individual may be taking steps back and forth without being able to reach the desired combination of genes in a reasonable time.

Further analysis of the algorithm performance revealed that this hypothesis was correct, as can be observed in Figure 3.18. In the first 350 iterations, there is almost always a steady increase in utility because, in the beginning, it is easier to achieve higher fitness. As the algorithm hits iteration 400, the utility starts to decrease slowly until it reaches iteration 600. By then it starts increasing again until the iteration 800 where it starts to decrease again. It resembles a cycle in which the utility goes back and forth just like we hypothesized.

We also analysed population diversity at different stages of the population. It was considered that a converged allele translates into 95% of the population sharing the same value for a given gene [17]. In test II, after one hundred iterations, the percentage of allele genes was higher than 90%. This occurrence means that the population is converging in the initial stages of the algorithm, which reduces population diversity. Consequently, it also reduces the chances of getting increasingly better solutions.

In a general problem, premature convergence causes the solution to stagnate at a local optimum instead of the global optimum. When the solution is stuck at a suboptimal stage, it may be difficult

Figure 3.18: Variation if maximum utility per iteration in Test II of experiment III.



or impossible to encounter the optimal stage without diversity across the population. However, in this problem, this does not apply because even without population diversity, it will always be possible to reach the optimal solution. If all the individuals of the population shared the same values for each gene, the crossover process would be redundant, and the algorithm is entirely dependent on mutations. If this is the case in our problem, it would just be a matter of time until it reaches the desired solution. Thus not having population diversity does not preclude the search for the optimum solution, it only delays it. In this problem, premature convergence may be due to a combination of factors: low population size, aggressive fitness function, and flat mutation rate. If an individual had a considerably higher fitness of than its remaining peers, the chances that this individual is replicated more times in the next generation is increased, which contributes to premature convergence. With a low mutation rate, the individuals don't change much from their previous self, which results in smaller population diversity. Coupling these factors with a moderate population size, increases even more, the chances of an early convergence.

3.3.4 Experiment IV

In the previous section, two main issues were identified: early convergence and the fact that the utility was not systematically increasing but instead was iteratively increasing and decreasing. For these reasons, changes were made into the first approach.

The selection process was changed because the algorithm was not being capable of having a consistent increase in utility. As verified in the previous experiment, the fitness of the best individual was going back and forth after a few iterations. So if we managed to save the best n

individuals of the population to the next generation, we would guarantee that they would still be there (elitism). If some of these n individuals managed to get a better utility through crossover or mutation, then the algorithm has achieved a better utility value. On the other hand, if the individual's fitness decreases, there is always a fallback to the original individual that was placed directly in the next generation. This mechanism is a trial and error method because it tries to get a better solution and if it does not achieve it, it tries again because we saved the initial state intact, i.e., the n best individuals of the population. Of course, by removing the n worst individuals of the population, we are discarding some genetic information, which reduces the population's diversity.

To counteract premature convergence, the mutation rate is set to a high value. This measure would generate individuals quite different from the ones that were saved for the next generation. Consequently, the population maintains a broad genetic pool across iterations. Moreover, an initial high mutation rate also may increase utility quicker because the initial set of individuals are not expected to be very good, given the size of the search space. This is true because the population is being created randomly, and we are disregarding heuristic approaches for the initial population's creation.

Since the mutation rate is set to a high value, there is the danger of the algorithm performing a random search, rather than a search guided by the quality of the previous solutions. So if the algorithm doesn't find a fitter chromosome in i iterations, the mutation rate drops by a factor of r. And it keeps dropping until eventually reaches a minimum value for mutation. As the mutation rate drops, the genetic diversity across the population is expected to decline as well. And finally, we turn from an exploration strategy to an exploitation strategy. In the initial iterations of the algorithm, we have a high mutation rate. Thus we are in an exploration phase because there are many genes of the individuals changing at once. In later iterations, we are in an exploitation phase because the mutation rate is lower, which causes a decrease in the population's diversity. As the individuals are increasingly close to each other, the search for an optimal solution becomes a local search because it is assumed that the current best solution only needs some of refinement to reach the corresponding optimum. The parameters i, r are the ones that regulate the trade-off that happens in the algorithm between exploitation and exploration.

This mutation operator is inspired in simulated annealing. Without going into too much detail, it is an algorithm that aims to find the global optimum in the space search. It consists of changing the solution, possibly accepting worst solutions based on a random draw. And the probability of accepting worse solutions is progressively changed towards zero.

Moreover, we also increased the population size due to low population diversity. Using Approach II, we performed the same tests as in the previous experiment. The parameters for the tests are represented in Table 3.5.

The distribution of weights after Test I is shown in Figure 3.19. It is possible to see that when using Approach II, we were able to obtain the best possible distribution in under a thousand iterations. All the weights from the middle region are set to zero while there are non-zero weights at each end of the distribution, which was what we were expecting. Comparing this result with the corresponding test from the previous experiment, we conclude that this approach behaved better

Parameters used for tests		
Parameter	Test I	Test II
Training set size	550	550
Population size	50	50
Population retaining rate (selection)	10%	10%
Crossover rate	75%	75%
Initial mutation rate	40%	60%
Number of iterations	1000	INF
K	20	20
Mutation values	[0;1]	[0;20]
Minimal mutation rate	0,01%	0,01%
Minimal number of iterations for decaying mutation rate	10	10
Mutation decaying ratio (r)	25%	25%

Table 3.5: Parameters used to run Test I and Test II of experiment IV.

as it was capable of reaching its goal in the same number of iterations.

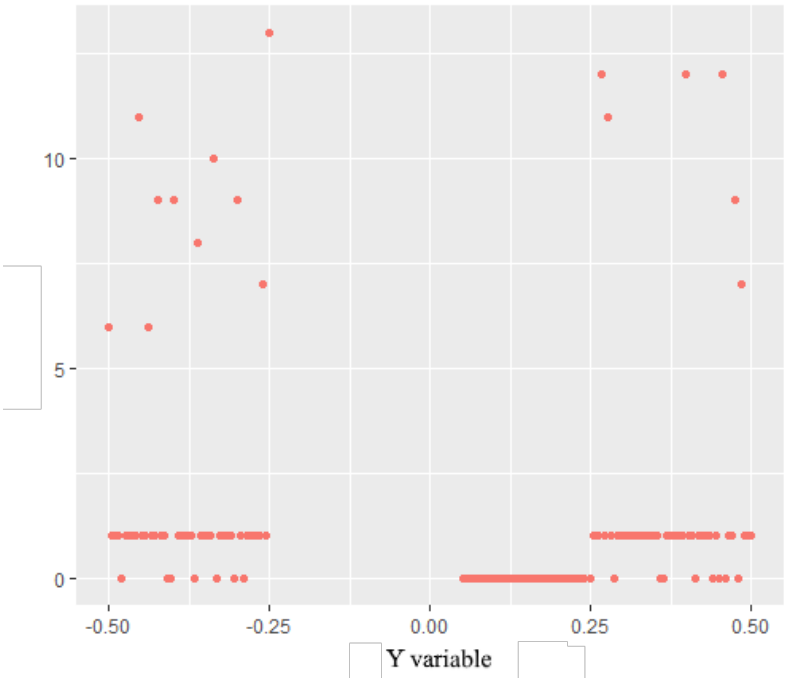
After successfully achieving the goal for Test I, the mutation range values can be extended again to $[0;20]$ for Test II. The goal for this test is to check if the algorithm is capable of finding the optimal solution and how much time does it take if the maximum number of iterations is set to infinite. Figure 3.20 shows the sequence of weight distributions for different iterations. The plot represents the state of the execution with a difference of 1000 iterations.

It took 19,000 iterations to achieve the perfect distribution of weights. Comparing the first 5000 iterations, it is clear that the weight's distribution is following a pattern. The higher weight of variable Y from the middle region are the first ones to be reduced. This reduction pattern happens because these examples are the ones that most hurt the utility function. What is also possible to notice is that the values of variable Y of the middle region that are closer to zero have non-zero weight. Just like we had discovered from experiment III, replacing the importance of these examples to zero would decrease the utility value. Another visible occurrence is that the weights of the extreme regions are increasing throughout the iterations. In the end, almost all of these examples have an influence of 20. Even though the best solution does not require that all the instances from the extreme regions have a weight of 20, it does increase the utility value between iterations. If we increase the relevance of these examples, it increases the number of instances from those regions in the new training set and thus forces the linear regression algorithm to adapt to those values, which leads to increased utility.

Concluding this experiment, it is possible to say that this Approach II is working moderately better than the Approach I for these types of problems. We were able to find the best model on Test I in under 1000 iterations while in the corresponding test from Section 3.3.3 that did not happen. Moreover, it was possible to find the best solution in Test II without replacing the initial population with a percentage of zeroes. Analyzing how Approach II performs throughout its iterations, it is concluded that the algorithm is moving towards the best solution, which is the end goal.

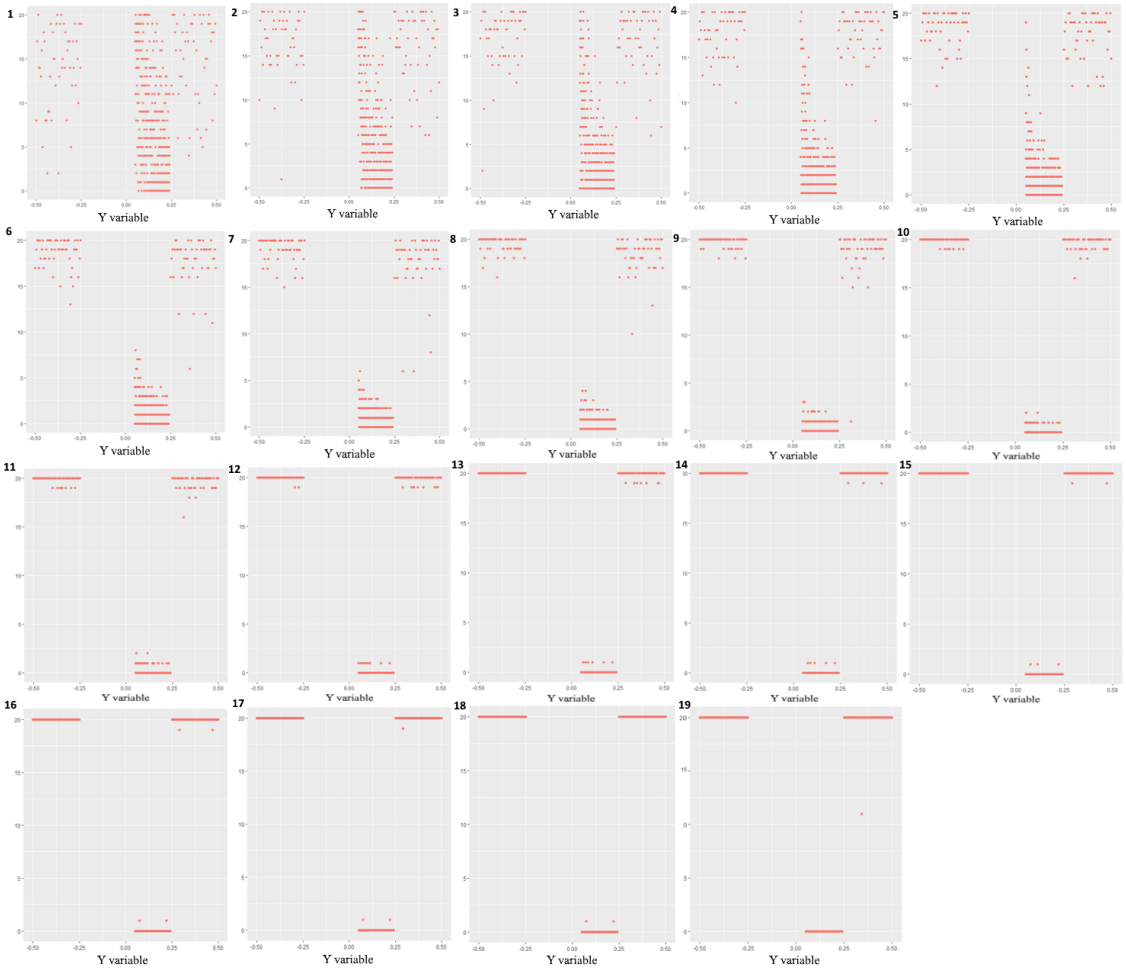
However, this Approach II must be used with care. If the parameters are incorrectly set, it

Figure 3.19: Distribution of weights obtained for the training set in Test I of experiment IV.



could mean that the exploration stage is minimal, and it all depends on the refinement of a local solution, i.e., the exploitation stage. This lack of exploration could mean that when the algorithm reaches the exploitation stage, the population is located in a local optimal and won't be able to achieve the global optimum. In our test, we used aggressive parameters for the exploration phase, i.e., setting the minimum number of iterations for the decaying mutation rate to 10. This setup was possible because even if the mutation rate sharply decreases, which diminished exploration, it would always be possible to reach the best solution.

Figure 3.20: Sequence of distribution of weights obtained for the training set in Test II of experiment IV separated by 1000 iterations.



Chapter 4

Empirical Study

This chapter presents the empirical analysis of results obtained in two different scenarios. In the first scenario, we used UCI data sets. In the second one, we used data provided by INOVRETAIL.

4.1 Benchmark data sets

This section discusses the experimental setup, results. It also details all data sets collected on the UCI website. The goal is to compare how the two approaches perform on real data sets with the same ML algorithm using the original training set(i.e. without reweighting). It is expected that both methods perform better than the ML algorithm, with Approach II also beating Approach I.

4.1.1 Experimental setup

This section details how the experiments were performed and how the data was collected. For these experiments, we used UCI data sets [1] for regression problems. The chosen data sets are Concrete, Housing, Orders, Traffic, Servo, Processor, Students, Airfoil, and Network. These data sets were chosen due to the similarity of the distribution of the target variable with a normal distribution and their small number of instances. The distribution of the target is important because the utility function we are using focuses on the extreme parts of the target's variable domain. These values are rare and, in many application, the most relevant ones. Regarding the number of instances in a data set, we opted for a low number of cases since this approach is computationally heavy. A summary of the data sets can be seen in table 4.1.

Each data set was randomly split to form a training set with 75% of the instances and a testing set with the remaining 25%.

Regarding the utility evaluation, the same process used in Chapter 3 is applied here to determine the utility function. Thus it is given more importance to the extreme regions rather than the middle. In these data sets, different values for the K parameter of the utility function are used to evaluate how different utility functions affect the performance of the algorithm. The values for the K parameter of the utility function are 20, 60, and 100.

[1] https://archive.ics.uci.edu/ml/datasets.php

Table 4.1: Information regarding used data sets.

Data set	# Examples	# Features
Orders	60	12
Traffic	135	17
Servo	167	4
Processor	209	9
Students	395	32
Housing	414	6
Network	630	9
Concrete	1030	8
Airfoil	1504	5

For these tests, the initial population used is created randomly with values ranging from 5 to 20. These values have a mean of 10 and a standard deviation of 3. The population size for Approach I is 10, and for Approach II is 50. The number of individuals chosen by the elitist mechanism is equivalent to 10% of the population, and the crossover probability is 75%. The mutation rate for Approach I is fixed at 0.2% while for the second approach it varies between 60% and 0.01%. The speed at which the mutation probability drops depends on the time it takes to compute the models per iterations. Models that take longer to compute, due to the high number of instances and features, will have an increased rate of mutation variation so that we can obtain results in a reasonable time.

Furthermore, the ML algorithms used were Decision Tree and SVM.

4.1.2 Results and discussion

The results obtained for DT and SVM are presented in Table 4.3 and 4.4 respectively. These tables compare utility and RMSE values between the two approaches developed and the ML algorithm in use.

One of the first things to notice is small utility values. These small utility values are expected unless the model was almost perfect across the domain of the target variable. The error of an example from the testing set is multiplied by its importance. The evaluation metric consists of dividing 1 by the sum of these multiplications, which results in low utility values.

When comparing either approach with the ML algorithm used for each test, it is possible to verify that in 100% of the tests, the utility value is higher using the developed approaches. This occurrence was the expected result because these approaches were developed to take into account utility while the ML algorithm doesn't consider it, but instead tries to minimize RMSE.

By comparing the results of different K values for the same data set with the same algorithm, it is noticeable that the more significant K is, the lower its utility is. This result was expected because the higher K, the higher is the importance given to the extreme examples. Following the same logic explained above, having a greater K implies multiplying the errors by a more significant value. It is not expected that the error decreases with increasing K but the utility should be lower..

Somewhat surprisingly, the RMSE is generally lower when using the proposed approaches than when using the ML algorithm directly on the training data. Given that the algorithms attempt o minimize error, we expect that they would achieve worse results in terms of utility but better results in terms of error. The reason for this may be that the rare extreme values have a higher chance of being harder to predict and thus have a high error. Since our approaches give more focus to extreme regions, they have a higher chance of reducing the error on such examples. This would lead our approaches obtaining both better utility and lower error.

When comparing the proposed approaches, the second one usually achieves the better utility value. This behavior is expected as the second approach makes a more extensive search than the first one. Therefore more models are generated, so the probability of finding a fitter model is higher using the latter. Moreover, Approach II was developed to fix some limitations of Approach I. However, Approach II is not guaranteed to perform better than Approach I as can be verified in the test results. This difference in performance has happened mainly due to incorrect parameterization. If in Approach II, the parameter i has a low value, it has little chance of finding a better utility so it is further reduced. Since in this approach, the minimum mutation probability is 0.01%, when it reaches this stage, it is unlikely to occur any mutation due to the small training sample size. Therefore there may be many iterations without actually existing an attempt to increase utility, considering that at this stage, the algorithm is in the exploitation phase, and the population diversity is close to null. The evidence to support this hypothesis, is that in most of the cases where Approach I has a better utility than Approach II, the training sample size is small. And if the training set is small, the lowest value of mutation rate is almost meaningless, which leads to bad performance. To correct this parameterization problem, one needs to change the lowest mutation rate to a relevant value and to increase the i parameter, i.e., the number of iterations required for the mutation rate to drop, where there is not an increase in utility.

To prove this hypothesis, we tried to find appropriate parameters for the Servo data set. To model this data set we use the decision tree algorithm. Since Approach I obtained better utility values than Approach II for every K with the decision tree, obtaining a better model with improved parameters for Approach II would add evidence in favor of our hypothesis. The parameters are shown in Table 4.2. The most significant difference is the minimum mutation rate that was changed from 0.01 % to 0.8%. This change allows an average of 1 mutation per chromosome when it reaches that stage. As such, instead of wasting computation costs with a low mutation rate, where only very few chromosomes are affected, every individual has a higher chance of mutation and thus contributing to the search. The utility value obtained from this test was 0.0533 which is not only higher than the previous recorded value for Approach II, but is also higher than the utility obtained with Approach I, which was the main goal of this test.

Figure 4.1 shows the comparison of individual prediction errors between Approach I and the regression tree with the original training sample in the Processor data set. The red dots represent the errors when using our method, while the blue dots correspond to the error when using the original training set. This was one of the data sets with the largest improvement in utility achieved by the proposed approaches. Even though utility values are low, it is possible to notice that the

Parameter	Value
Population size	50
Population retaining rate (selection)	10%
Crossover rate	75%
Initial mutation rate	60%
Number of iterations	INF
K	20
Mutation values	[0;20]
Minimal mutation rate	0,8%
Minimal number of iterations for decaying mutation rate	100
Mutation decaying ratio (r)	10%

Table 4.2: Parameters used for data set Servo.

difference in errors for some examples is quite large. When evaluating extreme region examples, which is the focus of our approach, and more specifically, the two instances that are on the far right of the figure, we can see a relevant difference in errors. Our approach was able to predict with relatively high accuracy these cases, while the regression tree with the original training set was not able to.

Analyzing the distribution obtained in the models presented in Figure 4.2, the results are far from what would be expected. Since the utility function has a shape of a positive quadratic function, we give more importance to extreme regions rather than examples in the middle area. Therefore it would be expected that the weight distribution would take the form of this quadratic function. The closer to the extremes, the higher the weights and vice-versa. However, that is not always the case. Looking at plots A, B, and C, the weight distribution looks random without any noticeable pattern. Example D is the only example that shows some resemblance to the expected distribution, with lower weights in the middle region and higher weights in the extreme areas. In general, these results may also be due to the way data are split into training and test sets. Since extreme values are rare, it it possible that there are no such values in the test set, which means that there is no need to assign high weights to extreme values to obtain high utility.

Regarding the SVM model (case B and D) there are more examples with a zero weight when compared to the decision tree algorithm. This may indicate that the algorithm stopped too early and, thus, the distribution is not what was expected. Moreover, by manual manipulation of weight values on some results, it was possible to increase the utility value on some data sets. So, the difficulty in finding the best or even a reasonable solution (due to the size of the search space) may be another reason why the expected distribution is not achieved. Finally, since some importance, even if small, is given to examples with a target value near the median, which are likely easier to predict, this may conduct the algorithm to focus on these small gains. Thus, the search would be stuck in a local optimum, making it harder to obtain higher benefits in regions that are more valuable.

Further analysis was done to test if what would be the expected weight distribution would indeed contribute to the best model or close to it. The results are presented in Table 4.5. The

Figure 4.1: Comparison of predictions' errors between the model with the original training set and Approach I.

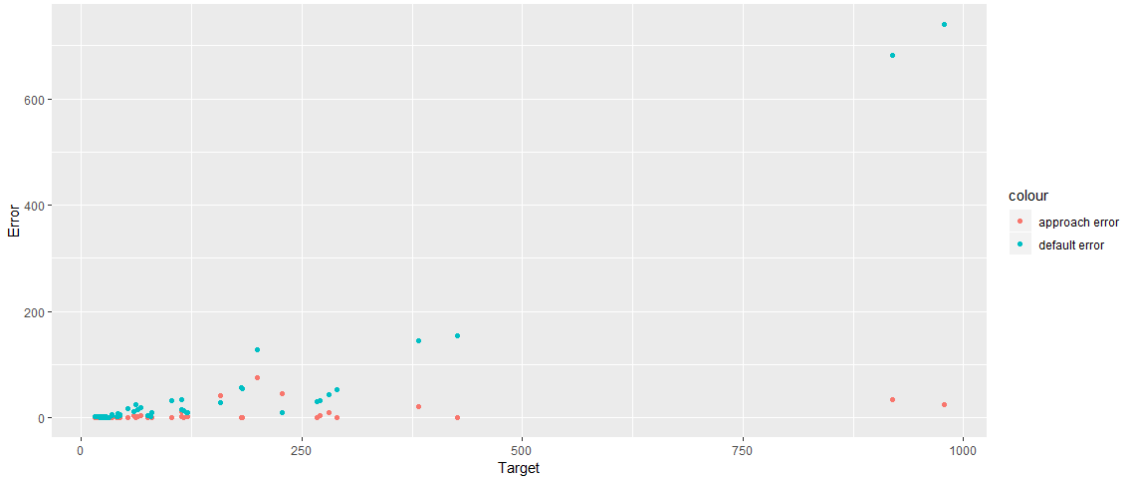


Figure 4.2: Distribution of weights on the training set. A) Housing training set using regression tree; B) Housing training set using SVM; C) Orders training set using regression tree; D) Orders training set using SVM;

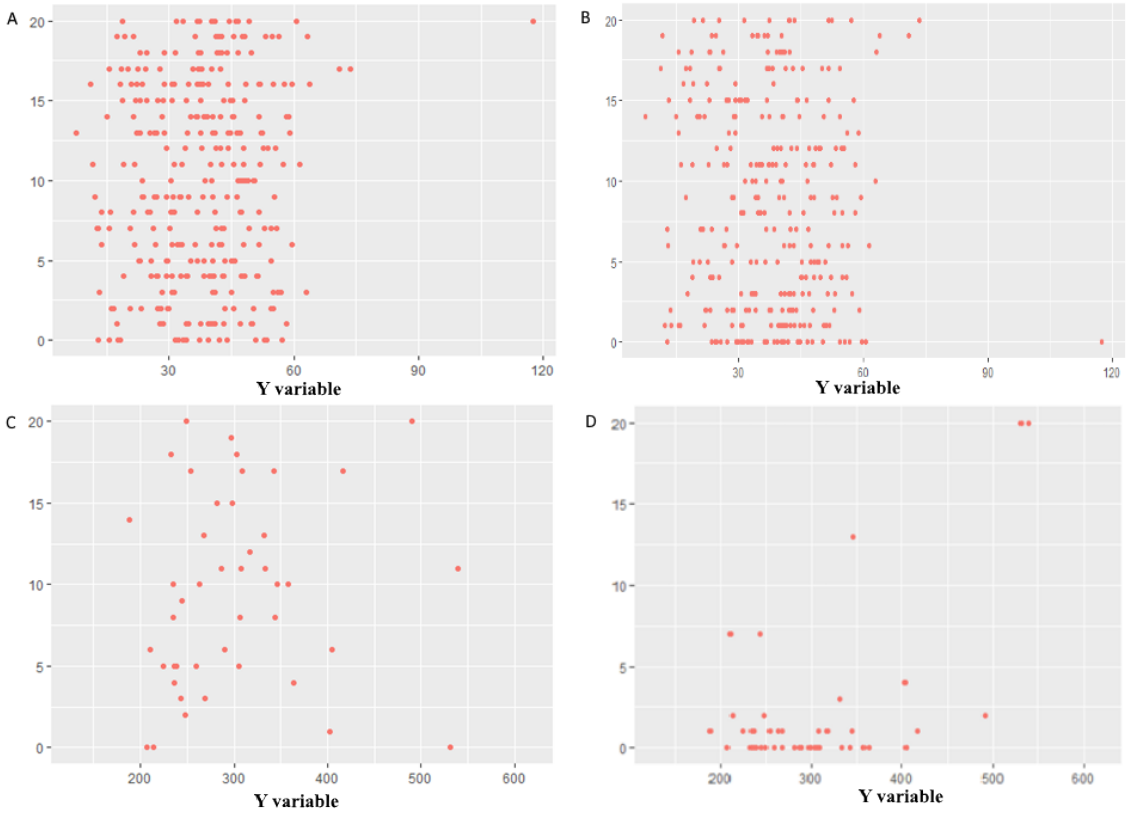


Table 4.3: Comparison of utility and RMSE values between the two approaches and the ML algorithm in use. In this case the it is used a decision tree for regression.

		Approach I		Approach II		ML algorithm	
Data set	K	Utility	RMSE	Utility	RMSE	Utility	RMSE
Concrete	20	0,00030	3,7	**0,00035**	**3,5**	0,00013	7,5
	60	0,00012	3,9	**0,00015**	**3,5**	0,00006	
	100	0,00008	**3,8**	**0,00010**	4,4	0,00003	
Housing	20	0,00070	6,0	**0,00074**	**5,8**	0,00043	8,0
	60	0,00031	**6,7**	**0,00035**	6,9	0,00020	
	100	0,00020	**6,4**	**0,00025**	8,7	0,00013	
Orders	20	0,00032	**28,1**	**0.00033**	28,3	0,00013	46,2
	60	**0,00011**	**27,4**	0.00011	31,4	0,00004	
	100	**0,00007**	**27,1**	0,00007	35,7	0,00003	
Traffic	20	**0,00368**	**1,9**	0,00353	2,6	0,00160	3,1
	60	0,00130	**1,9**	**0,00153**	2,7	0,00060	
	100	0,00075	**1,9**	**0,00098**	2,8	0,00037	
Servo	20	**0,03980**	**0,2**	0,02890	0,5	0,00917	0,6
	60	**0,02050**	**0,2**	0,01290	0,5	0,00305	
	100	**0,01348**	**0,2**	0.00780	0,7	0,00191	
Processor	20	**0,00072**	**15,1**	0,00048	21,3	0,00005	143,7
	60	**0,00032**	**13,0**	0,00022	26,9	0,00002	
	100	**0,00020**	**14,8**	0,00014	26,8	0,00001	
Students	20	0,00294	1,4	**0,00385**	**1,2**	0,00167	1,8
	60	0,00132	**1,2**	**0,00202**	1,3	0,00068	
	100	0,00076	1,3	**0,00123**	**1,2**	0,00043	
Airfoil	20	0,00037	1,9	**0,00048**	**1,9**	0,00020	3,4
	60	0,00016	**2,0**	**0,00019**	2,1	0,00008	
	100	0,00010	**2,0**	**0,00013**	2,2	0,00005	
Network	20	0,05130	**0,03**	**0,05532**	0,04	0,01805	0,09
	60	0,02351	**0,03**	**0,02403**	0,05	0,00769	
	100	**0,01606**	**0,03**	0,01469	0,05	0,00488	

Table 4.4: Comparison of utility and RMSE values between the two approaches and the ML algorithm in use. In this case the it is used SVM.

		Approach 1		Approach 2		ML algorithm	
Data set	K	Utility	RMSE	Utility	RMSE	Utility	RMSE
Housing	20	0,00063	6,1	**0,00067**	**5,9**	0,00038	9,3
	60	0,00033	6,2	**0,00034**	**5,9**	0,00018	
	100	0,00022	**6,1**	**0,00025**	6,4	0,00011	
Orders	20	0,00017	52,7	**0,00017**	**50,3**	0,00011	78,4
	60	0,00006	52,7	**0,00006**	**50,4**	0,00004	
	100	0,00003	52,8	**0,00004**	**50,3**	0,00002	
Traffic	20	0,00401	**2,3**	**0,00414**	2,3	0,00146	3,3
	60	**0,00164**	2,4	0,00163	**2,3**	0,00053	
	100	**0,00101**	2,4	0,00101	**2,4**	0,00033	
Servo	20	**0,03192**	**0,4**	0,03035	0,5	0,00531	0,9
	60	0,01437	0,6	**0,01523**	**0,5**	0,00201	
	100	0,01009	**0,5**	**0,01020**	0,5	0,00124	
Processor	20	0,00025	**23,8**	**0,00045**	41,9	0,00007	96,3
	60	0,00010	**23,4**	**0,00026**	40,4	0,00003	
	100	0,00006	**23,1**	**0,00018**	44,2	0,00002	
Students	20	0,00149	1,7	**0,00186**	**1,7**	0,00069	2,1
	60	0,00062	1,9	**0,00097**	**1,8**	0,00024	
	100	0,00039	**1,8**	**0,00051**	1,9	0,00015	

weights assigned to each example are calculated using the utility function. The maximum and the minimum weights, respectively. For every data set and every test value of K, there were only four times that the utility results were better than those obtained in the approaches. Moreover, these four times occurred in the same data set. These results demonstrate that using a heuristic function, such as used for these tests, to create a percentage of individuals of the population may lead to better and faster results. However, this is not a general rule. The obtained results mostly contradict what would be the expected behavior as the results are fairly worse. They are up to 55 times worse than using the developed approaches with random generation of the population.

In conclusion, it is possible to verify that with the developed approaches, we can obtain models that have a higher utility than the same ML with the original training set, as expected. Some data sets have regions located near the maximums or minimum of the target variable that are harder to predict. In these cases, the proposed approaches can be used to achieve a relevant gain in utility. Surprisingly, we achieve this without significant loss in the accuracy of the models. Furthermore, the expected weight distribution is not as expected. As this distribution appears to have no detectable pattern, it is not possible to understand clearly how these approaches are achieving their results. Further work is necessary to investigate this.

Table 4.5: Utility results obtained used the expected weight distribution on SVM and Decision Tree algorithms.

		ML algorithms		Comparison (Approach II / expected distribution)	
Data set	K	DT	SVM	DT	SVM
Concrete	20	0,00002	NA	17,50	NA
	60	0,00002	NA	7,50	NA
	100	0,00002	NA	5,00	NA
Housing	20	0,00004	0,00004	18,50	16,75
	60	0,00004	0,00003	8,75	11,33
	100	0,00003	0,00003	8,33	8,33
Orders	20	0,00026	0,00013	1,26	1,31
	60	**0,00022**	**0,00014**	0,50	0,43
	100	**0,00025**	**0,00014**	0,28	0,29
Traffic	20	0,00020	0,00023	17,65	18,00
	60	0,00022	0,00022	6,95	7,41
	100	0,00022	0,00022	4,45	4,60
Servo	20	0,00072	0,00065	40,14	46,69
	60	0,00082	0,00064	15,73	23,80
	100	0,00070	0,00059	11,14	17,29
Processor	20	0,00002	0,00002	24,00	22,50
	60	0,00002	0,00002	11,00	13,00
	100	0,00001	0,00002	14,00	9,00
Students	20	0,00018	0,00012	21,39	15,50
	60	0,00016	0,00011	12,63	8,82
	100	0,00018	0,00011	6,83	4,64
Airfoil	20	0,00005	NA	9,60	NA
	60	0,00005	NA	3,80	NA
	100	0,00005	NA	2,60	NA
Network	20	0,00190	NA	29,12	NA
	60	0,00207	NA	11,61	NA
	100	0,00175	NA	8,40	NA

Table 4.6: Information regarding tables provided by INOVRETAIL.

Tables	Attributes	Definition
Transaction	transaction_id, store_id, employee_id, date, hour, product, quantity and price	Each row contains how many times a product was purchased by a client on a store in a given date and hour. It also details the total payment and which employee performed the transaction.
Footfall	store_id, footfall, date and time	Each row represents how many people entered a store on a given day and time.
Target	store_id, date, target	Each row represents the target specified for a store in a given day.
Schedule	store_id, employee_id, date, time_start, time_end, shift_type	Each row represents the shift attributed to an employee of a store at a given date that starts in time_start and ends at time_end.

4.2 INOVRETAIL case study

In this section, we present the problem proposed by INOVRETAIL and how the developed approach can be used to address it. We also discuss the experimental setup and the results. The goal of these experiments is to verify how our approach performs in a real-world problem and how it fares against the predicted outcome of a retailer.

INOVRETAIL is a retail intelligence company that provides data science solutions to retailers. They focus on three different areas: sales incentives, customer experience, and improving in-store operations. The area that we focus on is sales incentives. The goal is to provide accurate sales targets to employees. A sales target is what is expected that a store would sale per unit of time and employee. If the employees can meet their goals, they are granted specific bonuses that reward their work. The closer the target is to the real sales potential, the more motivated the employees will be to reach that goal. However, different settings have different sales potential. There are periods where it is still possible to grow while in others, growth is an unlikely event. This is the problem proposed by INOVRETAIL. The task is to predict the daily targets for stores. The estimate must take into account the potential growth of a store.

4.2.1 Data analysis

This section explains the data made available by INOVRETAIL. Moreover, it also specifies the utility function used.

INOVRETAIL provided all the data used in this case study. It includes transactions, footfall, targets, and scheduling data. Table 4.6 describes these different types of information.

Regarding the footfall data, these numbers are recorded via infrared beams. These beams have some obvious limitations, such as when multiple people cross the beam line, only one person is counted. Furthermore, if its height is incorrectly set, it can miscount shopping trolleys or baby strollers. Therefore further attention is required when analyzing these values.

Cleaning the received data was the first step taken towards the creation of the data set. Some tables had repeated rows that needed to be removed. Secondly, footfall data were recorded hourly. Since the specified problem is to predict daily net sales values, it was required to group these values per day and store while adding the footfall counter to each row. The same transformation was done for transactions data, where we arranged all the transactions by day while summing the net sales and counting the number of transactions. With footfall and transactions data grouped by day, these two tables could be merged. The attributes after the merge are the date, footfall, net sales, number of transactions. These were the attributes provided by INOVRETAIL that were considered to be the most important to address this problem.

Now that all valuable attributes have been retrieved, the process of feature engineering begins. From the date, we can extrapolate four different fields: year, month, day, and weekday. Both weekday and month are divided into two fields. These two fields are meant to represent the cyclical properties of these features. The idea is to map the lowest value next to the largest one. For instance, regarding months, the goal is to place January next to December through data values. For this we make use of *cos* and *sin* functions in the following way: $var\_cos = cos(var * (2 * \pi / length(var)))$ and $var\_sin = sin(var * (2 * \pi / length(var)))$, where var is either the values attributed to month or weekday. For the variable month, these values range is $[0;11]$ while for weekday the range is $[0;6]$ and the length of these variables is 12 and 7 respectively. Lastly, we calculate the conversion ratio from the variables footfall and number of transactions: $conversionratio = number_of_transactions / footfall$. When calculating this variable, some values were higher than 1. For these examples, the conversion ratio was replaced by the mean.

Moreover, the footfall of these examples was also replaced by predictions using a regression model where the X variable was the net sales, and the goal was to predict footfall. This method was the chosen path because footfall and net sales were highly correlated variables. In the end, the data set had the following features: year, day, month_cos, month_sin, weekday_cos, weekday_sin, footfall, conversion_ratio, net sales.

The data provided includes seven stores from two different retailers. Details about these stores are presented in Table 4.7. Retailer 2 had more data from previous years when compared to Retailer 1. The reason for the different number of rows in stores with the same retailer is due to how Sundays and holidays are interpreted in different locations. For instance, some countries allow retailers to be open on Sundays while others restrict the number of Sundays a store can be accessible in a year.

4.2.2 Utility function

The way we adapt our approach to the problem at hand is by changing the utility function. In the previous experiments, the utility is measured by the domain of the target variable. The closer the target variable is to its minimum and maximum value, the higher its relevance is. However, in this problem, such cannot be applied. A different approach is needed. As such, it is considered that the only way we can increase the expected sales value is to change how operations are run in the

Table 4.7: Details about stores used in the experiment.

Store	Retailer	# Rows
Store 1	Retailer 1	675
Store 2	Retailer 1	645
Store 3	Retailer 1	636
Store 4	Retailer 2	1561
Store 5	Retailer 2	1574
Store 6	Retailer 2	1570
Store 7	Retailer 2	1556

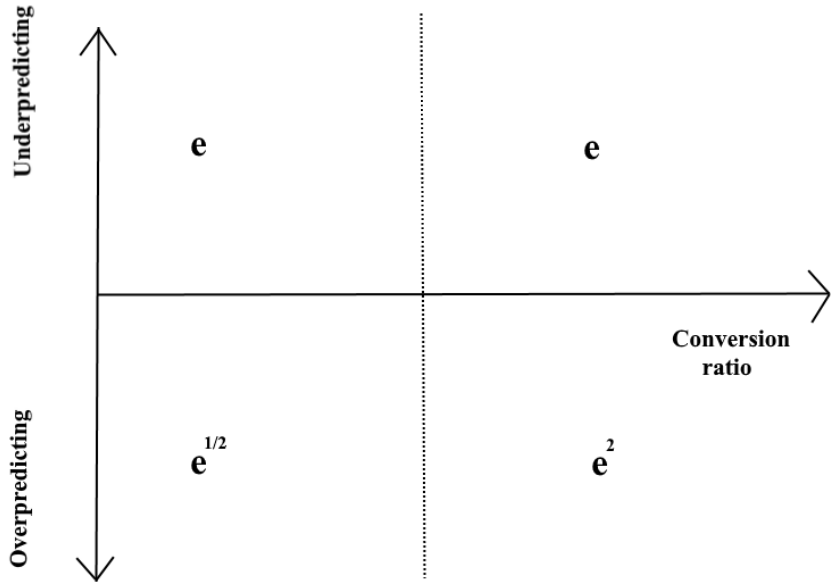
store. The following list details some examples of what kind of occurrences could lead to lower sales:

- An inadequate number of employees working in a shift;
- Bad staff recruitment. For instance, hiring people that lack social skills, technical knowledge, or cannot handle pressure may lead to low performance;
- Starting staff training during high traffic days may lead to lower sales values because the trainer has to focus on the trainee;
- Bad stock management.

These are all reasons that may cause lower sales in a day. Nonetheless, these reasons represent data which we have no access to. Therefore a performance measure is needed that can distinguish a bad from a good day in regards to store efficiency. The variable that was chosen to measure this performance was the conversion ratio. It represents the percentage of potential clients, i.e., people that walked into a store, that became clients, i.e. people that made a purchase. So if a store had a high conversion ratio, it means that the store is already performing well and an increase in sales is unlikely. On the other hand, if a store had a low conversion ratio, it means that the store is not performing well and an increase in sales is possible.

We propose a utility function that segregates the data into two parts , based on a threshold parameter. If a conversion ratio is higher than the threshold, the store is considered to have high performance. If it is lower, then it is considered to have insufficient performance. When predicting a determined value, there will likely be an error associated with the prediction when comparing to the real value. This error can be an overprediction or an underprediction. The idea is to penalize these types of error differently, depending on the class of conversion ratio. So, in the case of underachieving stores, an overprediction is less penalized than an underprediction. The reason is that if on a given day, the store is underperforming, it is likely that some problems are occurring that should be dealt with in the future to increase sales. So, it the problems are addressed, it is more plausible that in that day the store can easily increase their sales value. Thus an underprediction is penalized more than an overprediction. On the other hand, if a store already has a high performance on a given day, it is unlikely that it will still increase its net sales. This assumption is due

Figure 4.3: Representation of the utility function used for INOVRETAIL case study.



to the hypothesis that the store has reached its full potential. Therefore an overprediction is more penalized than an underprediction.

Figure 4.3 summarizes the utility function used. In this figure, the vertical dotted line represents the mean of the conversion ratio. The mean was the value that was selected to distinguish good and bad performance of a store in a day. As the mean is the average of all recorded conversion ratio, it is appropriate to use it as a threshold value. In this case, when the store is underperforming, overpredicting, and underpredicting by the same factor contribute to different utility values. In the case of an underprediction, the utility is the same as the error while overpredicting, the utility corresponds to the root square of the error. On the other hand, when the store has a good performance, overpredicting is more penalized than underpredicting by the same factor. So when underpredicting, the utility is the same as the error while overpredicting, the utility is the square of the error.

4.2.3 Experimental setup

This section details how the experiments were performed.

Regarding the experiments, two tests were performed for each store. In the first test, the real values of conversion ratio and footfall were used. The idea is that if we were in a hypothetical situation where we have a perfect model for predicting footfall and the number of transactions, what would we be able to achieve regarding net sales predictions? For the second test, we replaced the real values of the testing set with the best prediction from the following models: SVM with different kernel functions, random forest, and gradient boosted tree.

To evaluate these experiments, we compare the predictions obtained in the different tests between themselves and with the target variable that corresponds to the value stipulated by the store's manager. What is expected is that using the real values for the variables footfall and the number of transactions we would have a better fit than using predictions for these variables concerning utility and RMSE. When comparing both of these experiments with the targets, we would be expecting a better performance in utility with more certainty when using the real values instead of the predicted ones. Regarding the RMSE, the expectation is unknown because it depends on how the algorithm manages utility and how good the set targets are. Since overpredictions and underpredictions impact the evaluation in different ways using our utility function, it is not guaranteed that the RMSE should be smaller when using our approach.

The same time frame was used in all cases to split data into training and test sets. All instances ranging from the first of November of 2018 to the seventh of February were placed in the testing set, while every instanced that was recorded before these dates were placed in the training sample. This way, the testing set entails a time space similar to a quarter of a year which is a time frame that is commonly used for net sales prediction.

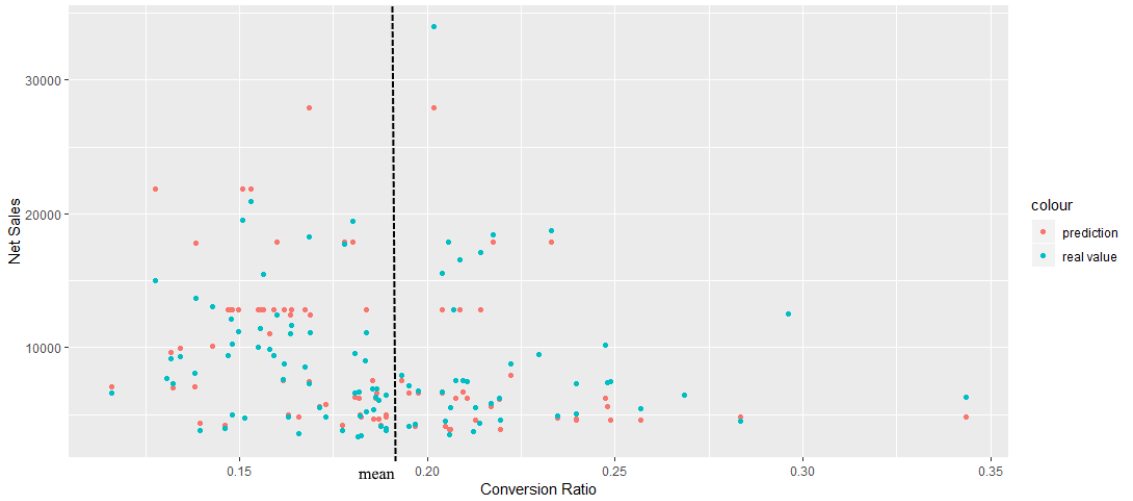
4.2.4 Results and discussion

In this section, the results obtained are presented and discussed. A comparison between the approach with real values of footfall and conversion rates as features and predicted values as features is detailed. In both cases, evaluation is done against the target variable and the store targets originally set by the business. An in-depth analysis is also performed on results obtained across the conversion ratio domain.

In general, when using our approach with real values for the features conversion ratio and footfall, we were able to obtain better utility results when compared with the same approach with predicted values for those features (Table 4.8). In the former, we have no errors in the values of two features which are expected to be important to the target results, and, thus, a model with better results is also expected.

However, this did not happen for Store 4, Store 5, Store 6, and Store 7 as the utility value was higher when using the predicted conversion ratio and footfall. Figures 4.4 and 4.5 show the comparison between the predictions and the real net sales values across the conversion ratio domain when using real and predicted values for the features conversion ratio and footfall for Store 7, respectively. The dashed vertical line represents the mean and distinguishes between days when the store underperforms (to the left) and overperforms (to the right). The figures are quite different regarding the conversion ratio value. When using the predicted values, almost all of the examples are at the left of the dashed line while when using real values, this pattern is not so clear. As we know, conversion ratio values lower than the mean is where overpredictions are encouraged. These type of predictions are the ones that represent the lesser cost throughout the domain of our data because the error is root squared. Also, since in Figure 4.5 most of the examples are overpredictions in underperformance areas, although the error is significant, the utility will not

Figure 4.4: Comparison between real net sales and predicted net sales of Store 7 using real values for conversion ration and footfall.



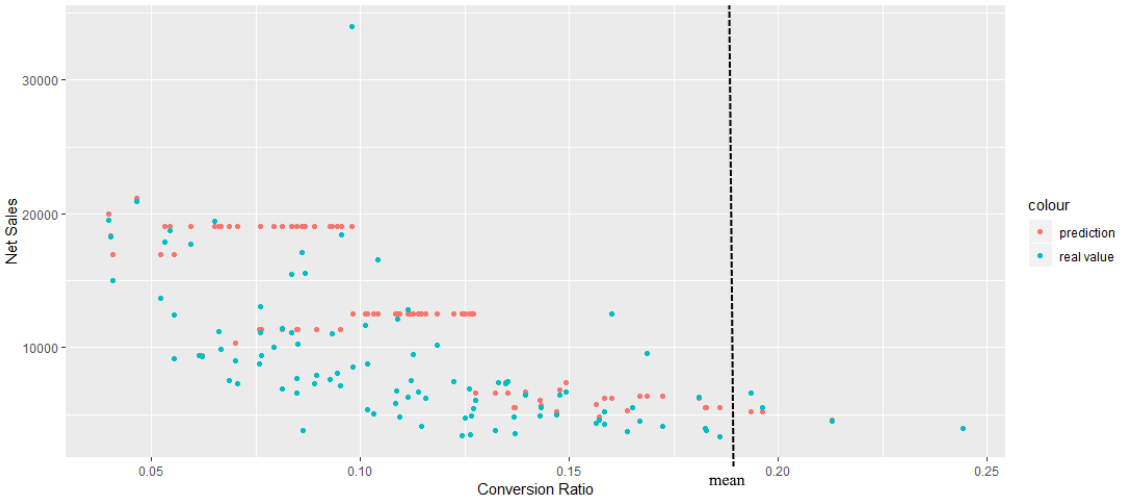
be significantly affected. So, this unexpected result is mostly due to inadequate modelling when predicting conversion ratio and footfall.

Regarding RMSE, when using the real values for the variables mentioned above, the error was lower when compared to the approach with the predicted values. Those two features are the most important ones when using a decision tree model. As such, when their values are more accurate, the error is lower as predicted.

When comparing each approach with the corresponding "baseline model", which corresponds to the regression tree model using the original training set, we observe that the utility is always higher when using our method. It is worth stressing that we managed to obtain a utility value, using approach II with the real values as features, that is 269 better than its counterpart model. Since regression trees optimize RMSE and not utility, these results come as no surprise. Regarding RMSE, it is not guaranteed that when using the developed approach, the results will be better. This behavior is due to the different penalizations on the error. Our goal is also to minimize the error but following a different approach specified in the utility measure. For instance, sometimes overpredicting by x may output a better utility than underpredicting by y in which $x \in]0; y^2]$. However, it may also generate a higher RMSE because the utility function allows that an overprediction may reach up to a square value of the error of an underprediction. In this case, the RMSE could be up to the square of the error associated with an underprediction.

Considering the targets stipulated by the store's manager, there is no guarantee that our method will output either a better utility value nor a better RMSE. This uncertainty regarding the expected results is because we did not know how accurate these targets would be when compared to the net sales value. If their error were minimal, even though they were not considering this utility metric, they would be able to achieve a better utility than with our approach. As stated before, the goal of the utility measure is also to reduce the error. If these targets are very accurate, then the error is

Figure 4.5: Comparison between real net sales and predicted net sales of Store 7 using predicted values for conversion ration and footfall.



small, and thus it will output a high utility value.

For every run test in each store, we were able to obtain a higher utility value than the implemented targets. Even some regression tree models with the original training set were able to outperform the targets with regards to utility. When analyzing RMSE, if we are using the real values as features, we obtained a better RMSE for almost every store. On the other hand, when using the predicted values as attributes, only one store was able to outperform the targets in this metric.

Figure 4.6 presents the comparison between the real net sales values and the predictions provided by Approach II with real values. The dashed vertical line represents the mean for the variable conversion ratio of the whole data set and not of the testing set alone. The red dots represent the predicted values, while the blue dots corresponds to the real ones.

There is a clear trend in the figure that is aligned with the utility function specified. The left side of the dashed line represents the area where there is an underperformance while on the right it corresponds to overperformance. As it was specified in the utility function, in underperformance's

Table 4.8: Utility results for each store with different setups.

Store	ML algorithm	Approach II	ML algorithm [1]	Approach II [1]	Target
Store 1	4,09E-07	0,00011	7,34E-09	6,49E-08	1,23E-08
Store 2	2,02E-07	5,77E-05	5,59E-08	6,56E-07	3,69E-08
Store 3	2,56E-07	8,97E-05	5,43E-08	5,01E-07	9,22E-08
Store 4	7,09E-09	2,54E-06	1,49E-08	1,80E-07	1,49E-08
Store 5	1,90E-07	2,88E-05	3,56E-07	5,40E-06	2,94E-07
Store 6	7,73E-08	4,66E-06	1,06E-07	3,76E-06	2,22E-08
Store 7	3,64E-08	3,36E-06	9.14E07	2.70E-05	2,86E-08

[1] Conversion ratio and footfall values used in this test are predicted by the author.

Table 4.9: RMSE results for each store with different setups.

Store	ML algorithm	Approach II	ML algorithm [1]	Approach II [1]	Target
Store 1	1357,76	1409,67	2326,97	2264,89	2571,03
Store 2	1203,48	607,84	1621,51	1811,16	1259,68
Store 3	922,31	598,72	1614,88	1507,49	1319,96
Store 4	2560,07	1934,69	4960,95	4575,58	2415,81
Store 5	1349,63	719,20	2633,88	1907,36	1461,50
Store 6	1473,83	1200,51	2208,32	1938,43	1245,06
Store 7	2410,97	2119,75	4372,25	5468,66	1767,04

[1] Conversion ratio and footfall values used in this test are predicted by the author.

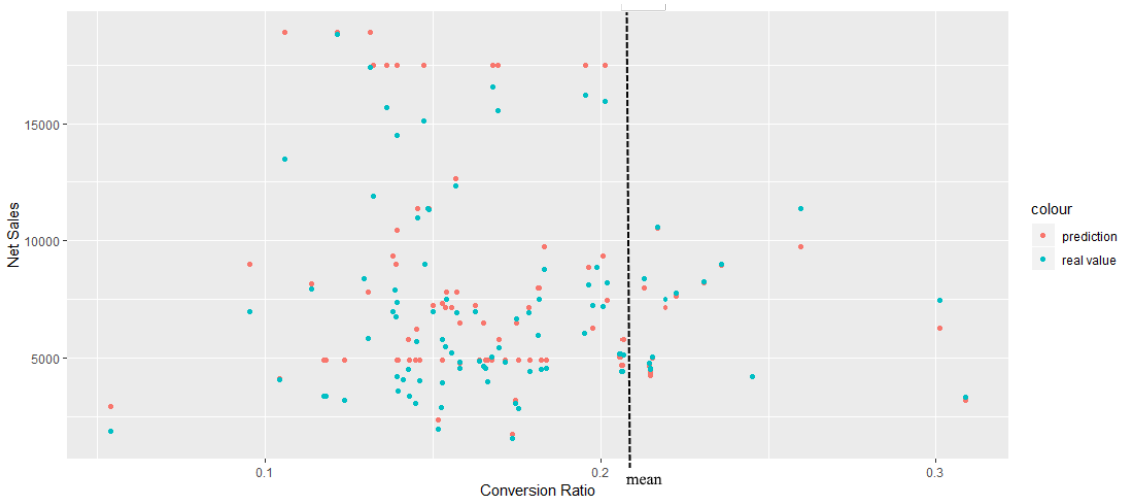
areas, an overprediction is less costly than an underprediction. As such, at the left of the traced line, most examples represent an overprediction. On the other hand, in overperformance's regions, an overprediction is more costly than an underprediction. Therefore, most examples on the area represent an underprediction.

The result obtained was the expected behavior, although it can not be guaranteed. For instance, there may be cases in underperformance's areas where an underprediction can still outweigh an overprediction. Let us consider that the best value an overprediction error can get for an example is 100. If the best value of an underprediction for the same example is 9, it outweighs the overprediction. This behavior happens because when considering $9^2 < 100$, it provides a better utility to make an underprediction.

In conclusion, the developed approach was able to improve the utility results when compared to the regression tree model with the original training set, and the targets. Considering RMSE, we were only able to obtain a better result with one store using the predicted values for modeling. However, when using the real values of footfall and conversion rate as features, we had improvements in almost every store with regards to this metric. This indicates that with better models of footfall and conversion ratio, our method would achieve a better utility and RMSE than the targets that were proposed by the business.

Moreover, it is expected that the overpredictions and underpredictions behave as explained before. So we can say with a reasonable degree of confidence that the approach is trying to optimize the specified utility measure, which is its goal.

Figure 4.6: Comparison between real net sales and predicted net sales of Store 1.



Chapter 5

Conclusions and Future Work

In real-world situations, the most valuable decisions may be taken regarding some areas of a data set. Usually, ML algorithms are not prepared to handle non-uniform costs across the data domain, as it covers the most common values. Although there is already some research on classification problems, there is a lack of work concerning regression problems.

In this thesis, we propose a new approach for utility based regression. In its core, it is a sampling method. We use a genetic algorithm as a means to vary the weight of a training set and optimize it according to a fitness function. This function is our utility measure. So what the algorithm does, is changing the weights of the training sample for each individual while optimizing a utility function. Regarding the case study in retail, the goal was to develop an utility measure that can be used on misclassification costs that take into account the store's net sales potential.

We developed two similar approaches, where the second one is a generally better approach than the first one as it fixes some of its limitations. These approaches were tested with artificial data, UCI data sets, and a case study in retail. Although we only tested these approaches for regression problems, they can also be applied to classification. The only required change is a suitable utility function.

We tested our approach using artificial data and the results obtained show that the algorithm works as expected. However, some modifications to the genetic operators were made to get improved results and surpass some encountered limitations.

Regarding UCI data sets, we were able to get better utility results in all the data sets tested when using our approach in comparison to the ML algorithm with the original training set. It was expected that the weight distribution would be similar to the utility function, i.e. higher weights in extreme rare regions and lower weight in central common areas. However, when analyzing the weight distribution of different training samples throughout its domain, that was not what was verified. This hypothesis was only correct for one data set out of nine. In all other data sets, our approach was able to obtain better utility results with different weight distribution. Therefore, our results indicate that the proposed approaches work even if their behaviour is not quite as expected.

For our case study in retail, we were able to achieve better utility results for every one of the seven stores tested when compared to targets defined by store managers.

The main limitation of this approach is the time it takes to have significant results. Modelling a high number of individuals per iteration is computationally costly. One way to overcome this problem is by using a fast ML algorithm that does not scale too bad with the number of instances or changing the approach's parameters. However, this latter method may result in suboptimal solutions. As such, this approach should not be considered for real-time data mining or when the model takes too much time to compute.

5.1 Future Work

Several directions for future work can be identified. One of the proposed adjustments is changing genetic operators functionality to a more greedy-like behaviour. For instance, the mutation could be changed into an operator that takes into account the utility differences obtained with previous mutations on the same gene. Thus, if the utility changes were positive, the mutation would promote similar changes to that specific gene. Conversely, if the change leads to worst utility, the mutation should try a different kind of change. This can not only lead to better results, but it can also lead to a better understanding of the importance of each example in a training set. A systematic analysis of these changes could lead to the the discovery of patterns in different data sets that could also be applied to others that are similar and also increase their utility result as well.

Appendix A

Experiment II results

This appendix presents some of the results from Experiment II detailed in Section 3.3.2.

Figure A.1: Results regarding Test II with 90% of zeroes per individual: A) Resulting model using our approach. B) Distribution of weights obtained in the training set.

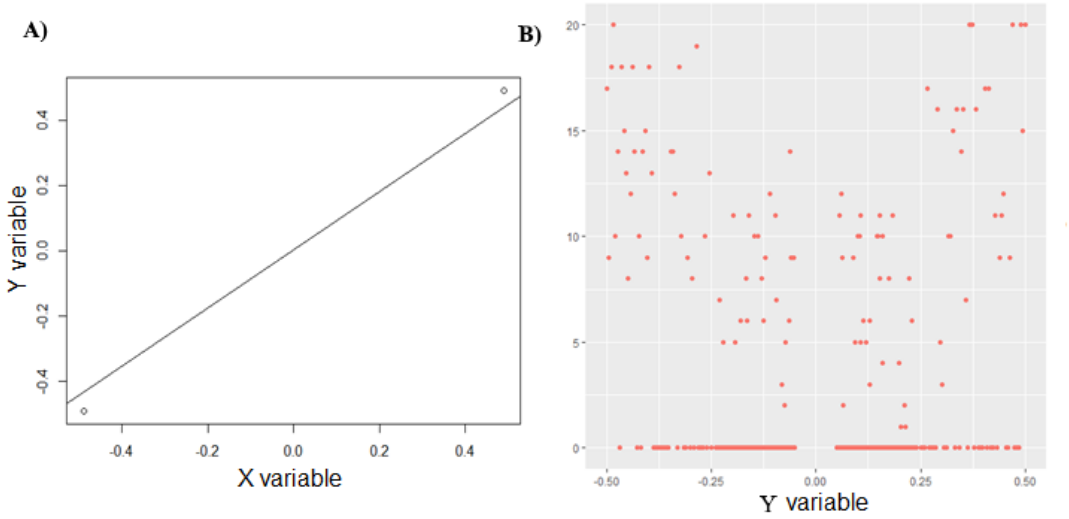


Figure A.2: Results regarding Test II with 80% of zeroes per individual: A) Resulting model using our approach. B) Distribution of weights obtained in the training set.

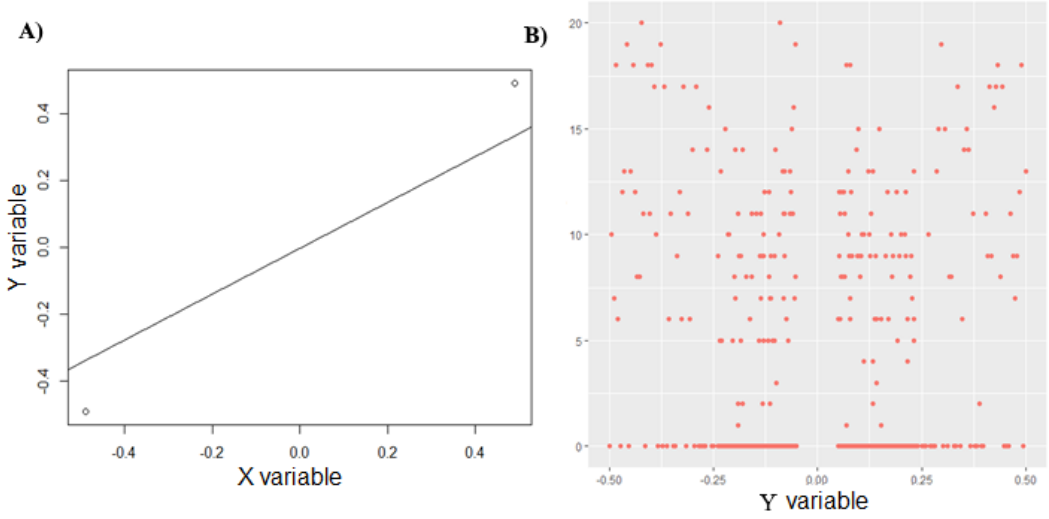
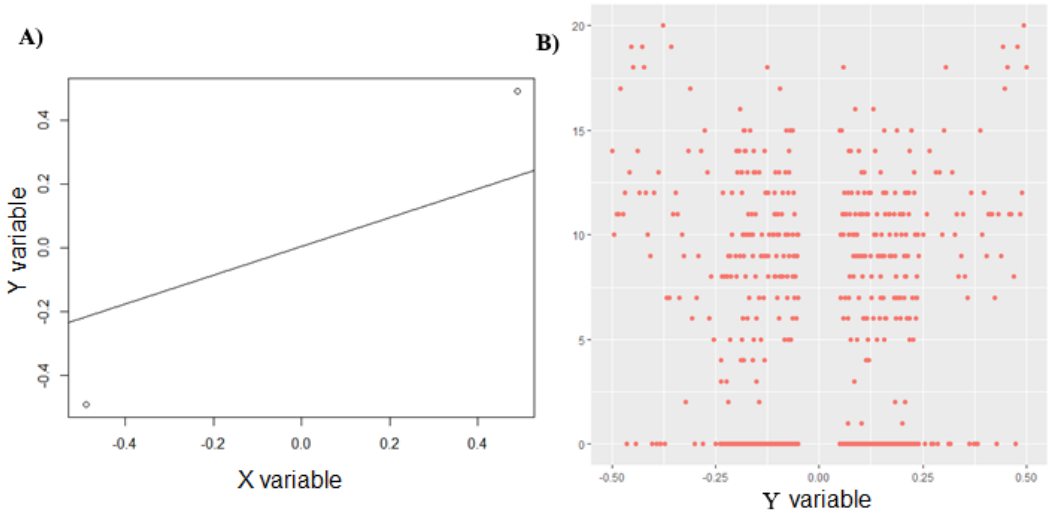


Figure A.3: Results regarding Test II with 60% of zeroes per individual: A) Resulting model using our approach. B) Distribution of weights obtained in the training set.



Appendix B

Experiment III results

This appendix presents some of the results from Experiment III detailed in Section 3.3.3.

Figure B.1: Results regarding Test I of experiment III with 80% of zeroes per individual: A) Resulting model using our approach. B) Distribution of weights obtained in the training set.

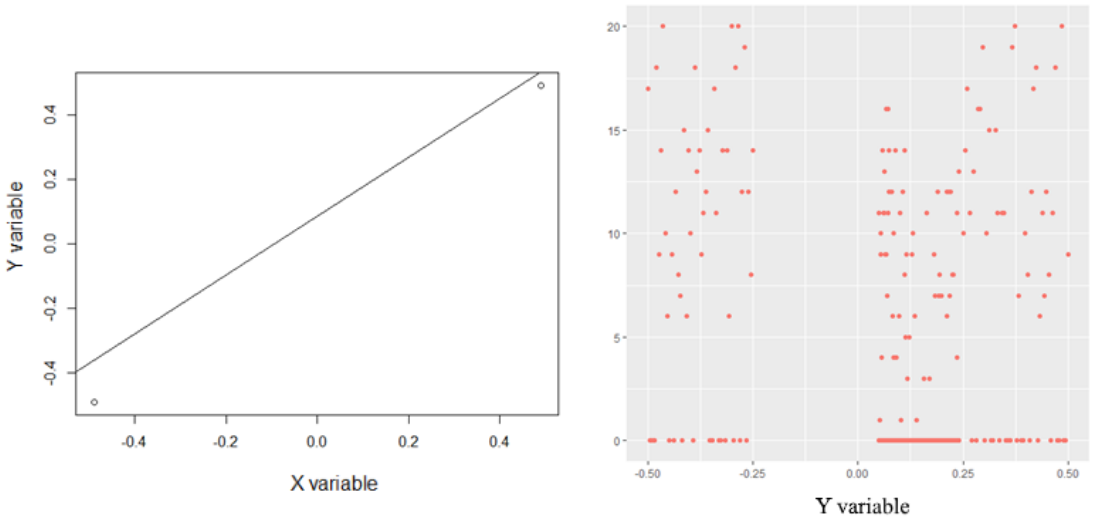
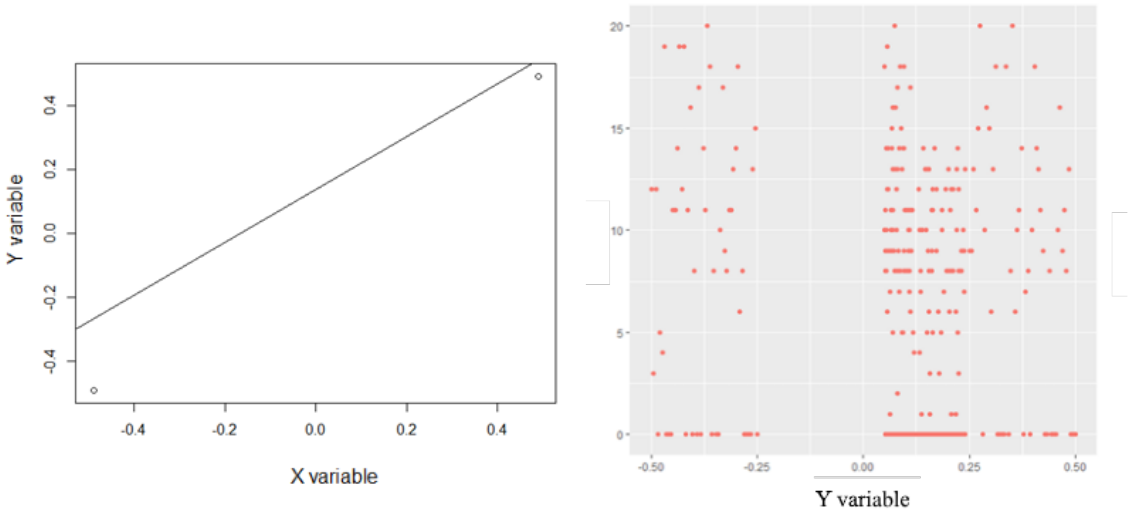


Figure B.2: Results regarding Test I of experiment III with 60% of zeroes per individual: A) Resulting model using our approach. B) Distribution of weights obtained in the training set.



References

[1] Paula Branco, Liaad Inesc Tec, Rita P Ribeiro, Liaad Inesc Tec, Eibe Frank, Bernhard Pfahringer, and Markus Michael Rau. Learning Through Utility Optimization in Regression Tasks. 2017.

[2] Paula Branco, Luis Torgo, and Rita Ribeiro. A Survey of Predictive Modelling under Imbalanced Distributions. 49(2):1–50, 2015.

[3] Deirdre B O Brien and Robert M Gray. Cost-sensitive Multi-class Classification from Probability Estimates. pages 712–719, 2008.

[4] Michael Brydon and Andrew Gemino. Classification trees and decision-analytic feedforward control : a case study from the video game industry. pages 317–342, 2008.

[5] Nitesh V Chawla. Chapter 40 DATA MINING FOR IMBALANCED DATASETS : AN OVERVIEW.

[6] Nitesh V Chawla, David A Cieslak, Lawrence O Hall, and Ajay Joshi. Automatically countering imbalance and its empirical relationship to cost. pages 225–252, 2008.

[7] Sven F Crone, Stefan Lessmann, and Robert Stahlbock. Utility based Data Mining for Time Series Analysis - Cost-sensitive Learning for Neural Network Predictors. pages 59–68, 2005.

[8] Pedro Domingos. MetaCost : A General Method for Making Classifiers. pages 155–164, 1999.

[9] Agoston E. Eiben and James E. Smith. *Introduction to Evolutionary Computing (Google eBook)*. 2003.

[10] Charles Elkan. The Foundations of Cost-Sensitive Learning The Foundations of Cost-Sensitive Learning. (May 2001), 2013.

[11] Charles Elkan. The Foundations of Cost-Sensitive Learning The Foundations of Cost-Sensitive Learning. (May 2001), 2013.

[12] Tom Fawcett. PRIE : a system for generating rulelists to maximize ROC performance. pages 207–224, 2008.

[13] George Forman. Quantifying counts and costs via classification. (April):164–206, 2008.

[14] David E. Goldberg. David E. Goldberg-Genetic Algorithms in Search, Optimization, and Machine Learning-Addison-Wesley Professional (1989).pdf, 1989.

[15] David E. Goldberg and Martin Pelikan. Bayesian Optimization Algorithm, Population Sizing, and Time to Convergence Martin Pelikan, David. E. Goldberg, and Erick Cantu-Paz. *Population (English Edition)*, (2000001):275–282, 2000.

[16] John H. Holland, E C Chu, and J E Beasley. Genetic Algorithms - Computer programs that "evolve" in ways that resemble natural selection can solve complex problems even their creators do not fully understand. *Scientific American*, 24(1):66–72, 1992.

[17] KA De Jong. An analysis of the behavior of a class of genetic adaptive systems, 1975, 1996.

[18] Fernando G Lobo. A Review of Adaptive Population Sizing Schemes in Genetic Algorithms. *Computer*, pages 228–234, 2005.

[19] Fernando G. Lobo and David E. Goldberg. The parameter-less genetic algorithm in practice. *Information Sciences*, 167(1-4):217–232, 2004.

[20] James G. March. Exploration and Exploitation in Organizational Learning. *Organization Science*, 2(1):71–87, 1991.

[21] Usama Mehboob, Junaid Qadir, Salman Ali, and Athanasios Vasilakos. *Genetic algorithms in wireless networking: techniques, applications, and issues*, volume 20. Springer Berlin Heidelberg, 2016.

[22] Alan Piszcz and Terence Soule. Genetic programming: optimal population sizes for varying complexity problems. *{GECCO 2006:} Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 1:451–456, 2006.

[23] Sam Ransbotham, David Kiron, and Pamela Kirk Prentice. Minding the Analytics Gap. *MIT Sloan Management Review*, 56(Spring, 2015):63–68, 2015.

[24] Rita Ribeiro. Utility-based Regression. 2011.

[25] Rita P. Ribeiro and Luís Torgo. Utility-based Performance Measures for Regression. *Booklet of the 3rd Workshop of Evaluation Methods in Machine Learning - held in conjunction with ICML'08*, 1:1–4, 2008.

[26] Lior Rokach, Lihi Naamani, and Armin Shmilovici. Pessimistic cost-sensitive active learning of decision trees for profit maximizing targeting campaigns. pages 283–316, 2008.

[27] Prithviraj Sen and Lise Getoor. Cost-sensitive learning with conditional Markov networks. pages 136–163, 2008.

[28] Anna Syberfeldt and Lars Persson. Using Heuristic Search for Initiating the Genetic Population in Simulation-Based Optimization of Vehicle Routing Problems. *Proceedings of Industrial Simulation Conference EUROSIS-ETI*, 2009.

[29] Kai Ming Ting. An Empirical Study of MetaCost Using Boosting Algorithms. pages 413–425, 2000.

[30] Luís Torgo, Rita P Ribeiro, Bernhard Pfahringer, and Paula Branco. LNAI 8154 - SMOTE for Regression. pages 378–389, 2013.

[31] Gary M Weiss and Ye Tian. Maximizing classifier utility when there are data acquisition and modeling costs. pages 253–282, 2008.

[32] Tian-Li Yu, Kumara Sastry, David E. Goldberg, and Martin Pelikan. Population sizing for entropy-based model building in discrete estimation of distribution algorithms. (2006006):601, 2007.