UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

EMBEDDED REGRESSION METHOD BASED ON DEMPSTER-SHAFER THEORY

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS,
MENCIÓN COMPUTACIÓN.

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN
COMPUTACIÓN

BELISARIO PANAY SCHWEIZER

PROFESOR GUÍA:
NELSON ANTRANIG BALOIAN TATARYAN
PROFESOR CO-GUÍA:
JOSÉ ALBERTO PINO URTUBIA

MIEMBROS DE LA COMISIÓN:
FELIPE JOSÉ BRAVO MÁRQUEZ
MAURICIO DAVID CERDA VILLABLANCA
CAROLINA ALEJANDRA BONACIC CASTRO

SANTIAGO DE CHILE
2020

Resumen

Los métodos de aprendizaje supervisado son cada vez más complejos y han ganado popularidad en variadas áreas de estudio. Métodos como las redes neuronales comúnmente son tratados como cajas negras, esto significa que se desconoce su funcionamiento interno por lo que se les trata como máquinas de entrada y salida. Estos métodos son preferidos por sobre métodos más simples debido a su fácil uso y buen rendimiento. Sin embargo, debido a su complejidad se desconocen las razones que determinan los resultados obtenidos. Esto ha generado problemas en áreas como la medicina, justicia y finanzas, donde vidas humanas se han visto perjudicadas. Aunque se ha dejado de lado el uso de métodos más simples por métodos más complejos, aún se requiere de éstos para aplicaciones de alto impacto.

Por esta razón, en este trabajo se desarrolla un método de regresión transparente basado en la teoría de Dempster-Shafer. Esta fue utilizada debido a su desempeño y simplicidad, donde una predicción puede ser explicada debido a la importancia o peso que tienen los casos que constituyen el conjunto de entrenamiento. El método computa una función de similitud de los vectores observados para producir una salida usando una distancia ponderada. La importancia de cada una de las dimensiones de la entrada es aprendida durante la fase de entrenamiento usando descenso de gradiente.

El método inicialmente fue puesto a prueba con datos sintéticos y conjuntos de datos conocidos, en donde se comparó su desempeño en tareas de selección de características y predicción. Los resultados obtenidos fueron comparables a los métodos de regresión más conocidos. Posteriormente, se utilizaron dos casos de estudio, en los cuales el método fue comparado con métodos del estado del arte. El primer caso fue el pronóstico de entradas de clientes en tiendas de *retail*. En éste se predijo un mes de entradas esperadas de clientes con resultados muy similares a los métodos aplicados comúnmente. El segundo caso estudiado en esta tesis fue la predicción de gastos médicos de pacientes en un hospital japonés. Para esto se utilizaron registros de pacientes en períodos anteriores. El método pudo detectar que para poder predecir los costos médicos es suficiente con solo usar los gastos anteriores de un paciente, como ha sido insinuado con anterioridad en la literatura.

Finalmente, se concluye la validez del método propuesto, aún siendo éste un método transparente. Esto debido a que una predicción puede ser seguida fácilmente mediante la importancia de las dimensiones de la entrada junto con la importancia o masa de cada uno de los elementos en el conjunto de entrenamiento. Los resultados presentados demuestran la capacidad del modelo, obteniendo desempeños comparables a los métodos usados en la literatura para los problemas estudiados.

Abstract

Supervised machine learning methods have become increasingly complex and have gained popularity in a variety of fields. In particular, methods such as neural networks are often treated as black boxes, meaning that their inner workings are unknown. Thus they are treated as input-output machines, and are preferred over simpler and more transparent methods because of their ease of use and outstanding performance. However, the reasoning behind their output is unknown due to their complexity. This has resulted in irreparable damage to human lives in fields such as medicine, justice and finance. Although complex methods have been preferred, transparent methods are still required in high-stake situations.

For this reason, in this work, a transparent regression method based on the Demspter-Shafer theory will be presented. Dempster-Shafer was used because of its performance and simplicity over other methods. A prediction is attributed to the importance of elements on the training set. The method uses a weighted distance to compute a similarity function of the observed vectors to produce an output. The weight of each dimension represents the importance of each feature, which is learned during the training phase of the algorithm using gradient descent.

Initially, the method was tested using synthetic data and well-known datasets, where its performance was compared in feature selection and prediction tasks. The results were similar to those obtained by the best well-known regression methods. Subsequently, two cases of study were used, where the method was compared to the methods used in the literature for these problems. The first case was the forecast of customer traffic in retail stores. In this problem, a month of the expected number of clients was predicted, with similar results to the methods commonly used for this problem. The second case studied in this thesis was the prediction of patient health care costs in a Japanese hospital. For this latter purpose, previous diagnosis, procedures, health checkups and billing information were used as input. The method found that it is enough using costs features in order to predict patients costs, as it has been implied by previous research.

Finally, the validity of the proposed method is concluded, keeping the goal of getting a fairly transparent method. This transparency occurs because a prediction can be easily followed by the importance of each feature in the input alongside the importance or mass of each element in the training set. The presented results show the capability of the method, getting similar performance to the best methods found in the literature for the studied problems.

A mis padres, Jose y Lily
Quienes me dieron mis principios y valores

Agradecimientos

Primero, quiero agradecer a mis padres Jose y Lily quienes me dieron mis principios y valores. Siempre remarcando lo importante que son los estudios en mi formación como persona. Además quiero agradecerles la paciencia que tuvieron con mis cambios de carrera y el tiempo que me tomó terminar esta etapa.

También quiero agradecer a mis hermanos Monse, Jose, Francisco y Ale por los momentos que pasamos. En especial a Monse por su ayuda con la redacción en español de este documento.

Quiero agradecer a los amigos que hice durante el Plan común, en especial a Coni y Jorge, quienes me ayudaron a aclarar las dudas que tuve durante la carrera y sin los cuales no hubiese sido posible disfrutar mi vida universitaria. También agradecer nuevamente a Coni por su ayuda con la redacción de este documento.

Igualmente agradecer a mis amigos de la especialidad: Americo, Jaev, Gabriel, Rodrigo, Pelao, Huan, Sergio y Joaquin por todos los buenos momentos. En especial a Joaquin y Sergio con quienes he aprendido bastante, tanto en el área profesional como en lo personal, gracias a la empresa que comenzamos juntos.

Un especial agradecimiento a mi profesor guía Nelson Baloian por toda su ayuda en este proceso. Por introducirme en el área de la investigación, y por darme la posibilidad de realizar la pasantía, la cual se transformó en este trabajo de investigación.

También quiero agradecer a Horacio Sanson y a todo el equipo de Allm, por toda su ayuda durante la pasantía y por hacerme sentir bienvenido durante mi estadía.

Agradecer también a mi profesor co-guia Jose A. Pino por su gran ayuda en el proceso de investigación resultante de este trabajo, además de darse el tiempo de enseñarme sobre mis errores de redacción. Enseñanzas que espero poder aplicar en el futuro.

Finalmente, quiero agradecer a los miembros de mi comisión Felipe Bravo, Mauricio Cerda y Carolina Bonacic por sus revisiones, las cuales me permitieron mejorar este trabajo.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The field of machine learning has progressed dramatically in the last years and it is one of today's most rapidly growing technical fields [1]. As machine learning improves, methods are becoming increasingly more complex. Particularly in deep learning, a sub-field of machine learning, methods have reached a point where humans can no longer understand the inner working of their algorithms. As a consequence, most current prediction models are considered black boxes, i.e, it is too complicated for a human being to comprehend the way the model performs its predictions. So it is considered as an input-output machine.

There has been a growing trend in applying these black boxes models for high-stakes prediction in areas such as healthcare, criminal justice and finance [2]. Unfortunately, the lack of transparency of these methods has resulted in severe consequences for people's life. There have been cases where people have incorrectly been denied parole [3] or their health risk has been incorrectly assessed because of their race [4].

In the last years there has been an interest in trying to explain the behavior of these black box methods (explanatory artificial intelligence) [5, 6, 7, 8, 9]. The most common approach has been having an external explainable model that works in parallel to the black box, which creates an explanation after each prediction; the explanation is computed for each prediction instance. Besides requiring an additional effort, these methods try to locally approximate the prediction, thus the explanations are not perfect. Moreover, the explanations become unreliable under high-stake situations.

Instead of using additional explanation methods for these situations it may be preferable to use inherently interpretable methods [2]. An interpretable method is a method that has the ability to explain or to present its results in understandable terms to a human being [10]. Interpretable models are in a range between fully transparent models and models that are lightly constrained in model form (such as models that are forced to increase as one of the variables increases, or models that, all else being equal, prefer variables that domain experts have identified as important) [2]. The relation between all variables are fully understood in fully transparent models, so it is possible to follow the prediction computation. A fully transparent method does not imply that is it fully interpretable at the same time, e.g., a linear regression is a transparent method, but if it has hundreds or thousands of features that contribute to a prediction the method, it is no longer interpretable.

Research towards finding more transparent methods has been overshadowed by explanatory artificial intelligence. This is especially true for supervised learning where the machine "learns" a function that maps an input to an output based on examples of input-output pairs [11]. There have been recent works towards developing inherently transparent methods. For example, Hu et al. [12] proposed an algorithm for optimal sparse Decision Trees for binary variables. Peñafiel et al. [13] proposed an interpretable classifier method based on Dempster-Shafer's Plausibility Theory, where a set of rules can be obtained for each class. Johansson et al. [14] presents an interpretable regression tree method using conformal prediction, which gives an interpretable confidence interval for its responses.

In this work, the focus will be on supervised learning. In particular for regression analysis, where a transparent regression method will be presented. The proposed method uses Evidential Regression [15], a transparent regression method based on the Dempster-Shafer Theory [16]. The method uses a k-Nearest Neighbors [17] approach with a weighted distance. It is able to rank the features of its input vector according to their importance for performing a prediction, thus providing understanding of the data.

1.2 Significance of the study

The main contribution of this work is a new transparent embedded regression method. The method is based on Evidential Regression [15] a regression method based on a fuzzy extension of the Dempster-Shafer theory [18] where the output is computed using a nonparametric, instance-based approach. The new method is capable of solving regression and feature selections tasks. Feature selection (FS) is the ability to select the features with the highest correlation to the target variable [19]; in particular they select a subset of features based on a certain evaluation criteria. Some features that do not meet the criteria are eliminated; the goal is to eliminate redundant or irrelevant features that are a *priori* unknown. This strategy decreases the number of dimensions, often resulting in an increase of models accuracy and time performance [20, 21]. During the learning phase, the method ranks features which increment the method prediction performance; thus the method is able to solve feature selection problems. Compared to other complex methods, the proposed method uses a k-Nearest Neighbors approach which makes it more intuitive and transparent.

The proposed method obtains comparable performances to well known embedded regression methods such as Random Forest and Gradient Boosting in prediction and feature selection tasks, while still being a fairly simple regression where prediction computation can still be understood. Moreover, as the proposed method is based on a generalization of the Bayesian Theory (Dempster-Shafer Theory) it its possible to obtain a confidence interval for each prediction.

1.3 Objectives

This section presents the objectives of this thesis project. Both general and specific objectives are explained.

1.3.1 Main Objective

The main objective of this thesis work is to create a new transparent embedded regression method based on Evidential Regression [15], which solves prediction and feature selection tasks, giving a confidence interval for the predicted value.

1.3.2 Specific Objectives

1. Acquiring knowledge about the state-of-the-art in feature selection and regression methods by literature research.
2. Extending the Evidential Regression principle in order to make it possible to rank features.
3. Obtaining data by testing the extended method in controlled and real cases.
4. Compare the accuracy of the method in real world problems, comparing it to state-of-the-art methods.

1.4 Expected Results

1. A validated transparent regressor method with an accuracy comparable to state-of-the-art methods.
2. A simple methodology for ranking the input features according to their importance for every prediction made by this regressor.
3. A simple methodology for explaining reasons for a prediction using this regressor.

1.5 Thesis Structure

This work is organized as follows:

1. Chapter 2 presents an overview of the theoretical background of the algorithms and concepts used in this work.
2. Chapter 3 presents the proposed method. It first explains the algorithm this method uses and then it describes the improvements introduced in order to achieve the goals of this thesis.
3. Chapter 4 presents experiments performed with this algorithm in order to test its ability to do feature selection tasks and prediction performance on synthetic and well known datasets.
4. Chapter 5 presents an experiment with real world data (store traffic forecast) and compares its results against state-of-the-art methods.
5. Chapter 6 presents another real world experiment on prediction of health care cost and compares its results against state-of-the-art methods.

6. Finally, Chapter 7 summarizes the conclusions of this work and discusses some future research lines.

Chapter 2

Related Work

This chapter presents an overview of the theoretical background of the algorithms and concepts used in this thesis. First, it introduces the key concepts of supervised learning and then, the main ideas of interpretability, feature selection and Dempster-Shafer Theory are presented in order to understand the motivations of this thesis. This is followed by a brief explanation of the regression methods used throughout this work.

2.1 Supervised Learning

Supervised learning [22] is a machine learning technique which trains an algorithm or statistical model through examples; for each one of these examples the algorithm receives as input a set of values as a vector which characterizes it along with the expected output and the learning process tries to find a relationship between these variables. This is nowadays one of the most common techniques used in machine learning as it provides maximum flexibility and enables computer programs find patterns from the given data [23]. The examples in a supervised learning problem are called training set. Formally, the training set is defined as follows:

$$\mathcal{L} : \{\mathrm{e}_\mathrm{i} = (x_\mathrm{i}, y_\mathrm{i})\}_{\mathrm{i}=1}^{N}. \tag{2.1}$$

Where e_i is an element of the training set, x_i is the input vector of e_i and y_i is its output or target value. Formally Supervised Learning models receive an input vector X and a target value Y, then the algorithm creates a mapping function $\hat{Y} = f(X)$. The algorithms can be classified in two groups. In one group the number of parameters is fixed respecting the sample size (parametric); in the other one the number of parameters can grow with the sample size(non-parametric).

Supervised learning models can be further grouped in classification and regression models depending on the type of the target variable Y. If the target variable is a quality, such as "black" or "white", then the model finding the mapping function is called a classification model. On the other hand, if Y is a numeric quantity such as "height", then the model is called a regression model. This thesis focuses on the latter one, where the target variable is a continuous value. Time series and non time series regression problems will be presented to study the performance and usefulness of the proposed method. A time series is a set of observations, where each one is recorded at equally spaced points in time [24].

2.2 Gradient Descent

Most supervised learning methods have parameters that need to be learned during the training phase. These parameters are learned using an optimization of some sort. An optimization refers to the task of either maximizing or minimizing a target function $f(x)$ by finding an specific x value. In a supervised learning setting a loss function needs to be optimized. The loss function is a function $L(y, \hat{y})$ where y is the target variable and $\hat{y}$ is the predicted variable. One of the most common optimization for supervised learning algorithms is Gradient Descent [25].

Given a differentiable function $f(x)$, the derivative $\frac{\mathrm{d}f}{\mathrm{d}x}$ gives the slope of $f(x)$ at the point x. So it specifies how a small change in the input value changes the output of the function:

$$f(x+\varepsilon) \approx f(x) + \varepsilon\frac{\mathrm{d}f}{\mathrm{d}x}. \tag{2.2}$$

The derivative is therefore useful for searching the input value which maximizes or minimizes the function output because it shows the direction in which x should be changed in order to make a small improvement in $f(x)$ output. This technique is called gradient descent. In this work, two variants of the gradient descent algorithm are used which are described as follows.

2.2.1 Stochastic gradient descent

The algorithm requires as input the learning rate schedule $(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k)$, number of iterations and initial parameters vector θ. The algorithm is shown in Algorithm 1.

Algorithm 1 Stochastic gradient descent (SGD)

```
1: function SGD
2:     for $t \leftarrow (1, n)$ do
3:         Sample a minibatch of $m$ examples from the training set $\mathcal{L}$
4:         $\hat{g} \leftarrow \frac{1}{m} \nabla_\theta \sum_{\mathrm{i}} f_{\mathrm{i}}(\theta_{t-1})$
5:         $\theta \leftarrow \theta - \varepsilon_t \hat{g}$
6:     end for
7:     return $\theta$
8: end function
```

2.2.2 Adam

The algorithm requires the learning rate (step size) α, exponential decays for the moment estimates $(\beta_1, \beta_2) \in [0, 1)$, the number of iterations n, a small constant δ used for numerical stabilization and an initial parameter vector θ_0. The algorithm is shown in Algorithm 2.

Algorithm 2 Adam algorithm

```
1: function ADAM
2:     $m_0 \leftarrow 0$
3:     $v_0 \leftarrow 0$
4:     $\hat{w} \leftarrow w$
5:     for $t \leftarrow (1, n)$ do
6:         $g_t \leftarrow \bigtriangledown_\theta f_t(\theta_{t-1})$
7:         $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
8:         $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
9:         $\hat{m}_t \leftarrow \frac{m_t}{(1-\beta_1^t)}$
10:        $\hat{v}_t \leftarrow \frac{v_t}{(1-\beta_2^t)}$
11:        $\theta_t \leftarrow \theta_{t-1} - \frac{\alpha \cdot \hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}$
12:    end for
13:    return $\theta_t$
14: end function
```

2.3 Interpretability

In this section interpretability will be discussed. Interpretability is a domain specific notion [26, 27], so there is not a single definition. Miller [27] defines it as how well a human could understand the decisions in the given context, Peñafiel et al. [13] define it as the possibility a human user to track the decisions made by an instrument based on artificial intelligence. For Ribeiro et al. [6] it means the ability of an instrument to provide qualitative understanding between the input variables and the response.

For this work, we will understand interpretability as the the ability of a model to explain or present its results in understandable terms to a human being [10]. There are various degrees of interpretability depending on a method transparency or on the presented explanations. Transparency connotes some sense of understanding in the inner mechanisms of a method. Lipton [28] considered transparency at three levels. First at the level of the entire method (simulatability), the second at the level of individual components such as parameters (decomposability), and last at the level of the training algorithm (algorithmic transparency).

Simulatability refers to the ability of a human being to simulate the process of the model to produce a prediction. This is a subjective requirement, however the capacity of humans is limited. For example, a linear regression is considered a transparent method, but if it has hundreds or thousands of features as input to produce a prediction then the method will have low simulatability. Decomposability is the ability of each part of the model (e.g. input, parameters and calculations) to admit an intuitive explanation. For example, the parameters of a linear model could be described representing the strength of association between each feature and the output. Algorithmic transparency applies at the level of the learning algorithm. It can be proved that training will converge to a unique solution, even for previously unseen data. Because of the subjective nature of simulatability, the aim of this work is to create a transparent method accomplishing decomposability and algorithmic transparency. However, it should be noted that an inherently interpretable method needs to fulfill the three listed requirements. Some of the most well known transparent methods for regression tasks are Regression Trees (RT), Linear Regression (LR) and k-Nearest Neighbors (k-NN) regression.

Another approach for a method to be considered interpretable to a certain degree is its ability to provide explanations. These are post-hoc explanations, i.e. explanations made after the prediction is computed. They do not show the inner workings of the algorithms but provide useful information to the end users. Explanations can be text, visualizations, local explanations (explain prediction instances) or explanations by example. Post-hoc explanations can always be applied to transparent methods, for example feature importance on a RT where the features are more important if they appear higher in the tree. In the last years there has been an increasing interest in developing methods for post-hoc explanations for black box methods, which are typically the most accurate methods but lack transparency (explanatory artificial intelligence) [5, 6, 7, 8, 9]. The most common approach has been using model agnostic approaches. These are in general external inherently interpretable methods that work in parallel to the black box, create an explanation after each prediction approximating the interpretable method to the black box response. Rudin [2] argues that these explanations models cannot have perfect fidelity, because if they had then the inherently interpretable method used to explain the outcome should be used instead. In this work, we acknowledge this claim and only model specific interpretation will be considered.

Measuring interpretability is still an open problem in the literature [29]. In this work, interpretability will not be measured. The main aim of this work is to create a transparent method for regression tasks, whether the method is interpretable or not depends on the context of problem it is used on. But the method will have a high degree of transparency so that it is meant to be used in high-stake situations when interpretability is a requirement.

2.4 Feature Selection

As explained in the previous section, every part of a transparent method needs to admit an intuitive explanation. Consequently, one of the desired features for the proposed method is the ability to evaluate the importance of the features in its input to produce an output, which is the case for methods considered to be transparent, such us RT and LR. Feature selection (FS) is the ability to select the features with the highest correlation to the target variable [19]; in particular it selects a subset of features based on a certain evaluation criteria. Some features that do not meet the criteria are eliminated in order to get rid of redundant or irrelevant features that are a priori unknown. This strategy decreases the number of dimensions, often resulting in an increase of models accuracy and time performance. FS has been of great importance in areas such as DNA microarray analysis, text categorization and information retrieval [20, 21]. Nowadays, there are several datasets with high dimensionality. Reducing the number of features while maintaining the model performance has become indispensable. This strategy decreases the dimensionality of the Euclidean space, therefore preventing the curse of the dimensionality phenomenon that causes a drop on model accuracy [30].

FS methods can be classified in three types: filter, wrapper and embedded methods [31].

- **Filter methods**: are typically the simplest and cheapest to compute; they apply statistical techniques to test the correlation of each feature with the target variable: normally, a threshold is set to choose the most relevant features. The most well-known filter methods include Pearson's correlation coefficients [32] which assigns a value to two variables, between -1 and 1 representing the linear dependency between them. F-score [33] is another example, where the weighted harmonic mean of the test precision and recall is computed varying from 0 to 1.
- **Wrapper methods**: are expensive because they test multiple subsets of features in a prediction model and select the subset maximizing the performance of the prediction.
- **Embedded methods**: unlike the previous methods, embedded methods include the feature selection process as part of their learning phase. Common embedded methods include various types of regression tree algorithms. Some of the most popular ones are the Random Forest (RF) a method which uses multiple regression trees to reach an outcome and Gradient Boosting (GB), where weak regression trees are built, and in each boosting stage the tree which is most correlated with the negative gradient of the method is selected.

This thesis presents an embedded method. Other embedded methods used for benchmarking are Regression Tree (subsection 2.6.5), Random Forest (subsection 2.6.6), Gradient Boosting (subsection 2.6.7) and Weighted Nearest Neighbors (WkNN). WkNN [34] is an extension of the well known k-Nearest Neighbors regressor (subsection 2.6.2) that uses a Weighted distance function in the k-Nearest Neighbors model and a gradient descent as an optimization approach finds the optimal weight vector.

2.5 Dempster-Shafer theory

The proposed method will be based on the Dempster-Shafer theory, also known as the theory of belief function [16]. An introduction to this theory is presented in this section.

Let X be the set of all plausible outcomes of an event that needs to be predicted, this is called frame of discernment. A mass assignment function m is a function satisfying:

$$m : 2^X \to [0,1], \quad m(\phi) = 0, \quad \sum_{A \subseteq X} m(A) = 1. \tag{2.3}$$

Where 2^X is the combination of all possible states and A is each one of the states and ϕ is the empty or null state. The term $m(A)$ can be interpreted as the probability of getting precisely the outcome A, and not a subset of A. Multiple evidence sources expressed by their mass assignment functions of the same frame of discernment can be combined using the Dempster Rule (DR). Given two mass assignment functions m_1 and m_2, a new mass assignment function m_c can be constructed by the combination of the other two using the following formula:

$$m_c(A) = m_1(A) \oplus m_2(A) = \frac{1}{1-K} \sum_{B \cap C = A \neq \phi} m_1(B) m_2(C), \tag{2.4}$$

where K is a constant representing the degree of conflict between m_1 and m_2 and is given by the following expression:

$$K = \sum_{B \cap C = \phi} m_1(B) m_2(C). \tag{2.5}$$

Petit-Renaud and Denœux [15] introduced a regression analysis algorithm based on a fuzzy extension of belief functions, called Evidential Regression (EVREG). Given an input vector X, they predict a target variable y in the form of a collection of evidence associated with a mass of belief. This evidence can be fuzzy sets, numbers, or intervals, which are obtained from a training set based on a discount function that takes their distance to the input vector X and is pooled using the Dempster combination rule (Equation (2.4)). They showed that their methods work better than similar standard regression techniques such as the k-Nearest Neighbors using data of a simulated impact of a motorcycle with an obstacle.

The EVREG model has been used for predicting the time at which a system or a component will no longer perform its intended function (machinery prognostic) for industrial application. Niu and Yang [35] used the EVREG model to construct time series, whose prediction results are validated using condition monitoring data of a methane compressor to predict the degradation trend. They compared the results of the EVREG model with six statistical indexes; the result was that the EVREG model obtained the best performance. Baraldi et al. [36] used the model to estimate the remaining useful life of equipment. Their results have shown the effectiveness of the EVREG method for uncertainty treatment and its superiority over the Kernel Density Estimation and the Mean-Variance Estimation methods in terms of reliability and precision.

2.6 Regression Methods

This section presents a brief explanation of the regression methods used throughout this work. These methods are later used to benchmark the performance of the proposed method.

2.6.1 Seasonal Auto Regressive Integrated Moving Average

The Autoregressive Integrated Moving Average (ARIMA) model is often used as baseline for time series forecasting. The future value of a variable in this model is assumed to be a linear function of past observations. This thesis will deal with an extension of the basic ARIMA, called seasonal ARIMA (SARIMA) [37].

SARIMA is composed of two parts, a non-seasonal and seasonal model.The non-seasonal part of the method is defined by three variables (p, q, d) and the seasonal part by $(P, D, Q)_m$. P and p indicate the number of autoregressive terms (lags of the series). D and d are the differencing that must be done to stationarize the series. Q and q indicate the number of moving average terms; m indicates the seasonal length of the data.

2.6.2 K-Nearest Neighbors Regression

K-Nearest Neighbors (k-NN) Regression [38] is a non-parametric learning algorithm. It uses a set of observations that are closest to an input variable X to compute a prediction $\hat{Y}$.

Specifically, the k-NN fit for X is the mean function of its k-nearest neighbors in the training set $\mathcal{L}$. The closeness of a point is evaluated by a distance metric $||\cdot||$.

An improved version of this method called Weighted Nearest Neighbors (WkNN) will be used. The method uses a weighted average, where the contribution of each neighbor is proportional to its proximity [39, 34]. To compute the contribution of each neighbor it uses a weighted distance to compute the nearest neighbors. The weight vector is learned during training phase using gradient descent (section 2.2). In this work, k-NN regressor will refer to the WkNN regressor with an identity vector as weights.

2.6.3 Support Vector Regression

Support Vector Regression (SVR) is a learning machine model implementing the structural risk minimization inductive principle to obtain a good generalization on a limited number of learning patterns [40]. With a training set $\mathcal{L}$, the goal of the method is to predict the outcome value $\hat{Y}$ of every element of the training set that has at most ε deviation from the real target variable Y [41]. A prediction is calculated as:

$$\hat{Y}(X) = K(w, X) + b. \tag{2.6}$$

Where w is a vector in the same dimensional space as X and b is a constant, and K is a kernel function. In this thesis, two types of SVR will be used. The first is linear SVR which uses a SVR with a lineal kernel, where the kernel function is the dot product. And the non-linear SVR with a Gaussian Radial Basis function (RBF) kernel that is defined as:

$$K(w, X) = \mathrm{e}^{-\frac{\|X-w\|}{2\sigma^2}}. \tag{2.7}$$

The predictions need to be as flat as possible. This means that w needs to be as small as possible. One way to ensure this is to minimize the norm of w. However, sometimes this optimization is not feasible or errors need to be allowed. Vapnik [42] suggested the formulation of the optimization problem where the constant $C > 0$ determines the trade-off between the flatness of the mapping function and the amount up to which the deviation larger than ε are tolerated.

2.6.4 Gaussian Process Regression

A Gaussian Process (GP) [43] is a collection of random variables, where a finite number has a joint Gaussian distribution. A GP is completely defined by a mean $m(x)$ and covariance function $K(x, x')$ ($y \sim GP(m, K)$). Lets consider a training set $\mathcal{L}$ with X a set of inputs and Y the set of outputs, and a validation set with an input X_* and output Y_*. Since the key assumption in GP is that the data is represented by a Gaussian distribution, the data can be represented as:

$$\begin{bmatrix} Y \\ Y_* \end{bmatrix} = N\left(\begin{bmatrix} \mu \\ \mu_* \end{bmatrix}, \begin{bmatrix} \Sigma & \Sigma_* \\ \Sigma_*^T & \Sigma_{**} \end{bmatrix}\right). \tag{2.8}$$

Where μ and μ_* are the means of the training and validation set. Σ, Σ_* and Σ_{**} are the covariance of the training, training-validation and validation sets. Y is known so the target is to compute the value for a new observation Y_* given Y. The conditional distribution is defined as:

$$Y_*|Y \sim N(\mu_* + \Sigma_*^T \Sigma^{-1}(Y - \mu), \Sigma_{**} - \Sigma_*^T \Sigma^{-1} \Sigma_*). \tag{2.9}$$

This is a posterior distribution for a new observation. Then, the GP is defined as:

$$\begin{aligned} Y|\mathcal{L} &= GP(m_{\mathcal{L}}, K_{\mathcal{L}}), \\ m_{\mathcal{L}}(x) &= m(x) + \Sigma(X,x)^T \sigma^{-1}(Y - m), \\ K_{\mathcal{L}}(x,x') &= K(x,x') - \Sigma(X,x)^T \Sigma^{-1} \Sigma(X,x'). \end{aligned} \tag{2.10}$$

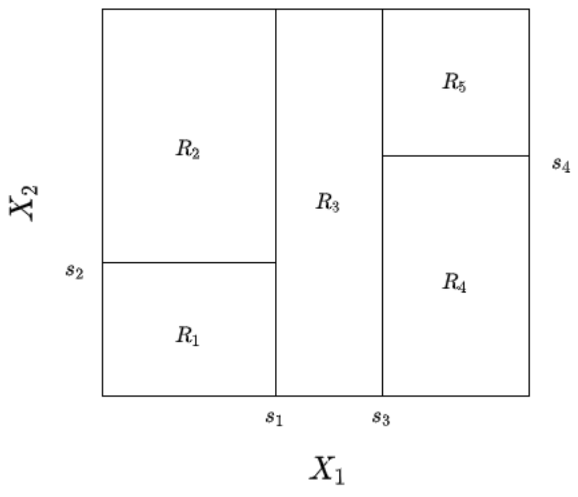
Where $\Sigma(X,x)$ is the covariance between each training sample and x. In a GP, it can be assumed that there is noise in the training set $\mathcal{L}$. An output can be predicted as:

$$Y(x) = g(x) + \varepsilon. \tag{2.11}$$

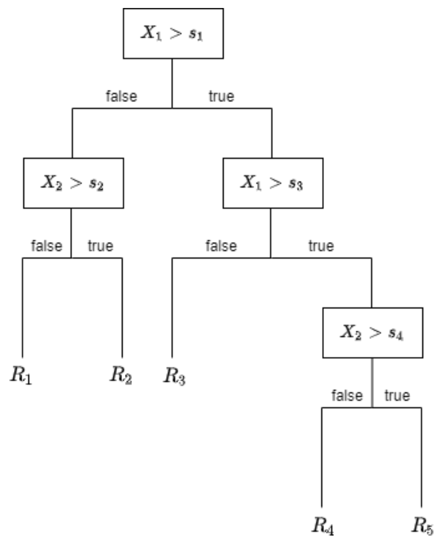
Where $\varepsilon \sim N(0, \sigma_n^2)$, $g \sim GP(m, K)$ and $Y \sim GP(m, K + \sigma_n^2 \delta_{\mathrm{ii'}})$. With $\delta_{\mathrm{ii'}} = 1$ if $\mathrm{i} = \mathrm{i'}$. So the covariance function for a noisy process is the sum of the covariance of the noise and the data.

2.6.5 Regression Tree

Regression Tree (RT) [38] is a method that partitions the feature space and then fits a simple model (e.g. a constant) in each partition. Let us consider a training set $\mathcal{L}$ of N inputs with an input set X and an output Y. The algorithm automatically decides on the features to split and the splitting points, and also what topology the tree should have. Suppose first that a partition of M regions $R_1, R_2, \ldots, R_m$ is given. An example RT in a two dimensional space is shown in Figure 2.1.



(a) Space partition.



(b) Tree partition.

Figure 2.1: Regression Tree space partition example.

The implementation used in this thesis defines a constant in each region to predict an outcome. To create these regions, a minimization criterion needs to be selected. In this work the Mean Squared Error (MSE) will be used (eq. 3.18) so that large errors are penalized. Finding the best binary partition in terms of the minimum mean squared error is generally computationally infeasible. Therefore, a greedy algorithm is normally used. This algorithm starts with the process of finding a split point s for one variable j which defines a pair of half-planes. The splitting point is found by minimizing the error criterion in each one of the planes. This process can again be repeated in each new region. The number of regions or the size of the tree in a RT is a tuning parameter. A very large tree would make the method to overfit the data, explaining the training set exactly. Commonly the tree size is set by stopping the splitting process when a given number of nodes is reached.

2.6.6 Random Forest

Random Forest (RF) [44] is an ensemble method, a method that uses multiple learning algorithms to obtain best performance. RF is a combination of RTs (subsection 2.6.5) where each tree depends on the value of a random vector sampled independently and with the same distribution for all trees in the forest. The output is computed as the average of each tree output. Each tree is grown and not pruned based on any error measure. This means that the variance of each one of these individual trees is high. However, by averaging the results this variance can be reduced without increasing the bias. An example RF is shown in Figure 2.2.

2.6.7 Gradient Boosting

Gradient Boosting (GB) [45] is an ensemble method, that uses a base learner (e.g. Linear, spline or tree), examine its residuals (difference between the predicted and the target variable), and fit a model based on the residuals around a loss function. This process is repeated

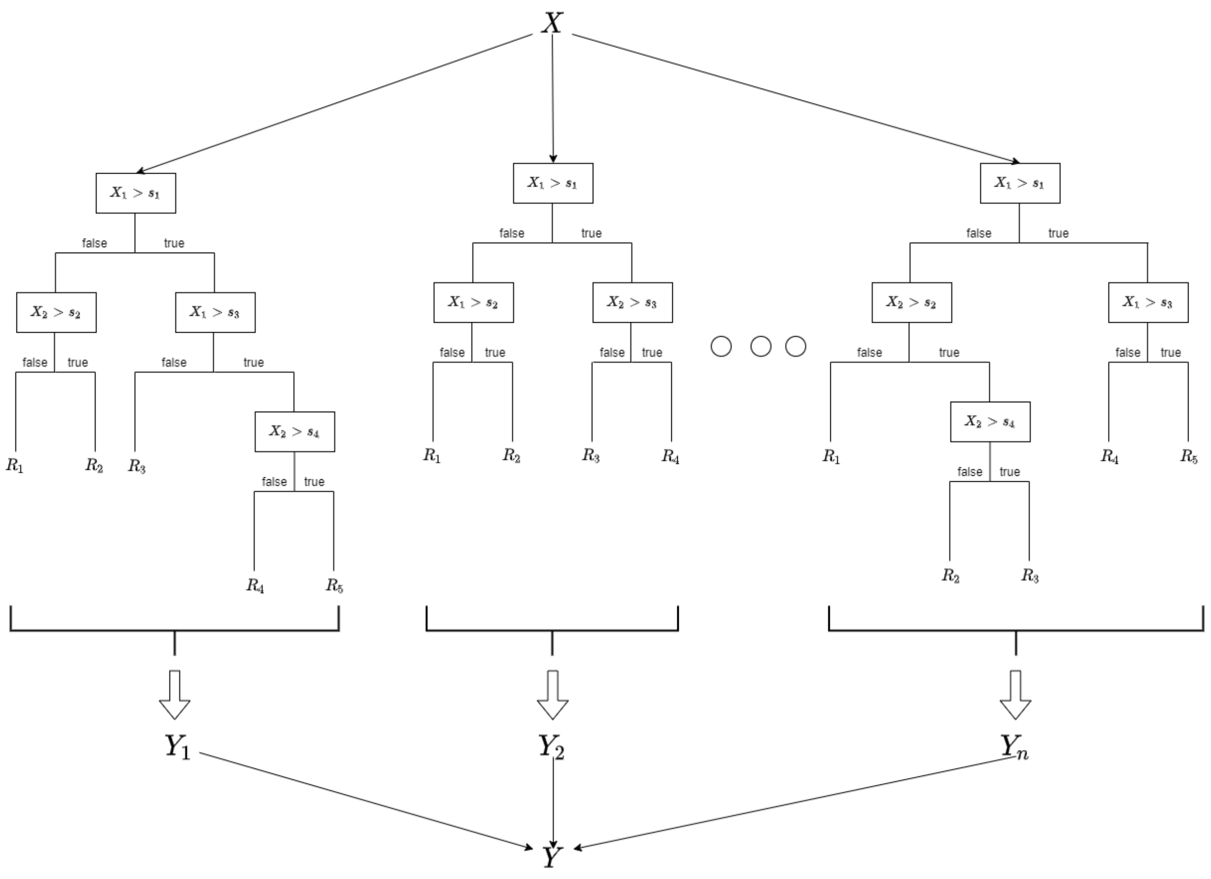


Figure 2.2: Example RF.

until it reaches a stopping criterion.

As mentioned above, GB can be applied to a range of different base learners. In this thesis, GB will refer to a method with an RT as its base learner. The principal difference between boosting methods and the methods presented so far, is that optimization is held out in the function space [46]. Given an specific loss function $L(Y, \hat{Y}(x))$, and a tree based learner $h(x, \theta)$, the solution of the parameters can be difficult to obtain. To deal with this problem, a new function $h(x, \theta_t)$ to be the most parallel to the negative gradient $\{g_t(x_\mathrm{i})\}_{\mathrm{i}=1}^{N}$ is selected.

2.6.8 Artificial Neural Network

An Artificial Neural Network (ANN) [47] is a black box that directly learns the internal relations of an unknown system, without guessing functions for describing cause-and-effect relationships. There are variety of architectures for ANNs; in this thesis an ANN will refer to a Multilayer Perceptron (MLP) neural network. The MLP is a fully connected feed forward network with one or more layers of perceptron units between the input and output layers. The example architecture of a MLP is shown in Figure 2.3.

In Figure 2.3, each circle represents a neuron and every line is a connection between them.

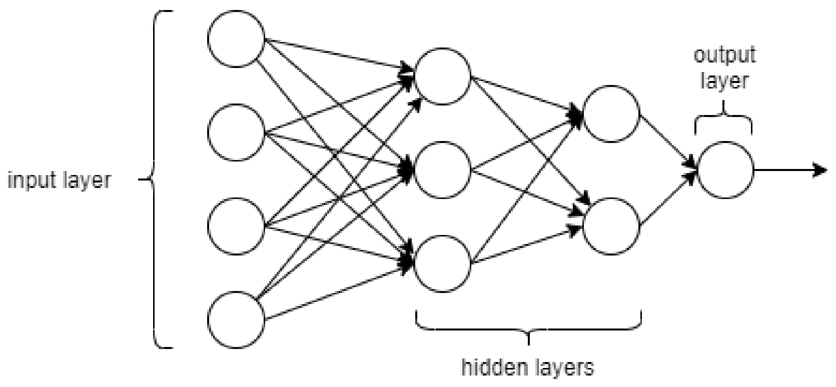


Figure 2.3: MLP example architecture.

The input flows from left to right. In each neuron the output is computed by an activation function as:

$$\sigma(X) = g\left(\sum_{\mathrm{i}=1}^{N} w_{\mathrm{i}} x_{\mathrm{i}} + b\right). \tag{2.12}$$

Where X is an input vector of size N, w and b are parameters of the neuron. And g is an activation function e.g. Sigmoid function. The parameters w and b of each neuron are learned during the training phase using Back Propagation (BP). BP is an algorithm that propagates the error backwards using a gradient descent 2.2 technique to minimize a cost function L between the predicted and the real values.

2.6.9 Long Term Short Memory Recurrent Neural Network

Recurrent Neural Networks (RNNs) [48] are a type of deep neural networks specifically designed for sequence modeling. In Figure 2.4, an unrolled representation of a RNN is shown.

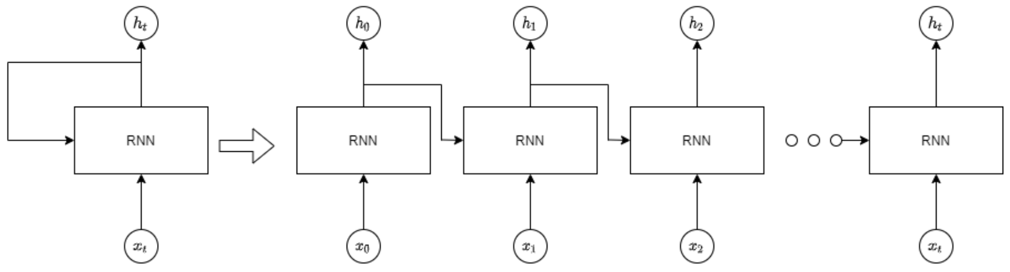


Figure 2.4: Unrolled RNN.

In Figure 2.4, the output h_t is used for the computation of h_{t+1}. However, traditional RNNs implementations suffer from the problem of vanishing gradients and thus have problems identifying long-term dependencies. The Long Term Short Memory (LSTM) [49] neural network was introduced to solve this problem.

LSTM recurrent neural network have "LSTM cells" that have an internal recurrence [25], in addition to the recurrence of the RNN. Each cell has the same inputs and outputs as an ordinary RNN, but with more parameters and a system of gating units that controls how much information is let through. LSTM uses three gates, the forget gate f controls if the memory cell is reset to 0, the input gate i controls whether the memory cell is updated and the output gate o controls whether the information of the current cell state is made visible. All the gates have a sigmoid activation function so they have a smooth curve that is differentiable. In Figure 2.5, a LSTM diagram is shown.

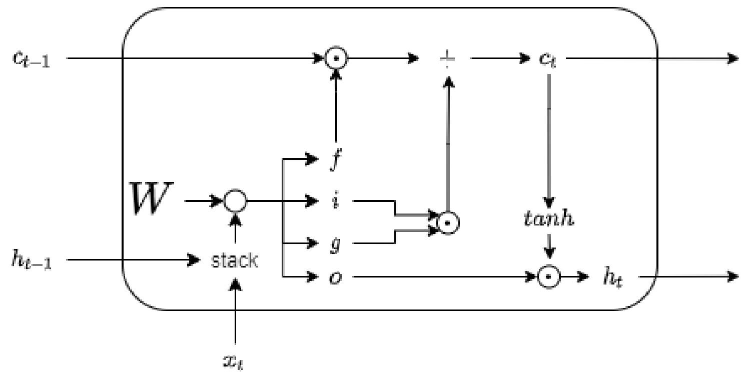


Figure 2.5: LSTM cell diagram.

Chapter 3

Model proposal

The aim of this work is to create a transparent regression method, with a performance comparable to state-of-the-art methods. This work is inspired by the work of Peñafiel et al. [13], where an interpretable classifier based on the Dempster–Shafer Theory (DST) [50] was presented. The Dempster-Shafer Theory (DST) [50] is a generalization of the Bayesian theory of subjective probability [51]. It is more expressive than classical Bayesian models since it allows us to assign "masses" to multiple outcomes measuring the degree of uncertainty of the process. So the outcomes produced by the method can be tracked to the pieces of evidence used by the method. The work of the interpretable classifier could be extended but as it is a pure classification algorithm, the output is assumed to be discrete, and some procedures do not apply to continuous outputs, e.g., the time complexity grows exponentially with the number of classes. Petit-Renaud and Denœux [15] introduced the use of Dempster–Shafer theory for regression problems; they presented a regression model that uses DST called Evidential Regression (EVREG) to find the expected value of a variable using a set of examples as evidence. EVREG will be used because of its transparency and performance. Each sample in the training set is assigned an importance value obtaining better results than other distance based algorithms as it was shown by Petit-Renaud and Denœux [15]. EVREG will be extended using a weighted distance and gradient descent. The weight of each feature will be updated with gradient descent, enabling the model to perform feature selection (FS) tasks. The weight of each feature will represent the importance of this feature to predict an outcome in a dataset; thus, a set of masses with these weights will be computed; the masses will represent the importance of samples in the training set to predict an outcome.

The EVREG [15] method uses a set of observations that have occurred in the past as evidence in order to predict the target value of a new observation. Each observation in this evidence set is given a mass which represents the similarity of the new observation with each observation in the set. This similarity is calculated using a distance function (e.g. Euclidean distance) in the feature space of the observations. The DST ensures that the masses of this evidence set add up to 1, so they can be easily transformed to a probability distribution. Then an expected value is computed as the mass of each past observation m_{i}, times its observed target value (y_{i}), as shown in 3.1.

$$E[y] = \sum_{\mathrm{i}=1}^{N} m_{\mathrm{i}} * y_{\mathrm{i}}. \tag{3.1}$$

DST is characterized for reasoning with uncertainty. EVREG uses the width of the observed target variable interval (difference between maximum and minimum target variables) to represent uncertainty as another piece of evidence. The importance given by the model to this interval represents the uncertainty the model has when predicting an outcome. For example, when the new observation is at a great distance from the evidence set, the model assigns a high value to the evidence of the variable y interval; thus a high uncertainty is obtained in the model outcome, resulting in upper and lower bounds for the predicted target variable proportional to the interval of the target variable observed in the evidence set.

EVREG has been used for predicting the time at which a system or a component will no longer perform its intended function (machinery prognosis) for industrial applications [35, 36]. Niu and Yang [35] used EVREG in a time series task, to predict the degradation trend of a methane compressor. They compared the results of the EVREG model with six statistical indexes; the results show the EVREG model had the best performance. Baraldi et al. [36] used the model to estimate the remaining useful life of industrial equipment. Their results have shown the effectiveness of the EVREG method for uncertainty treatment and its superiority over the Kernel Density Estimation and the Mean-Variance Estimation methods in terms of reliability and precision.

First, EVREG will be described (Section 3.1). Then, its time complexity and time improvement will be explained in Section 3.2. EVREG will be extended using gradient descent and a weighted distance function in Sections 3.3 and 3.4; the goal will be to improve the model regression performance and enable it to perform feature selection tasks, which will allow it to rank features in a space. Using this new distance function and optimization approach, the model will update the weight of each dimension in every iteration of the gradient descent algorithm. The extended model will be able to rule out or assign low weights to irrelevant features, and assign higher values to the ones that strongly influence the output. This is an important characteristic since feature selection has been shown to improve model accuracy and computing times [39, 52, 34]. The aim is that this enhanced EVREG method will perform as well as state-of-the-art regressors for tabular and time series regression problems. Also the model will possess a feature selection ability, ranking the features representing the correlation of each feature with the target variable, enabling its users to gain more knowledge of the used data.

3.1 Evidential Regression

EVREG [15] is a Supervised Learning algorithm. This kind of algorithms use a sample set as examples to make a prediction. The samples in a Supervised Learning problem are called training set. Formally, the training set is defined as:

$$\mathcal{L} : \{\mathrm{e}_{\mathrm{i}} = (x_{\mathrm{i}}, y_{\mathrm{i}})\}_{\mathrm{i}=1}^{N}. \tag{3.2}$$

Where $\mathrm{e_i}$ is an element of the training set, x_i is the input vector of $\mathrm{e_i}$ and y_i its output or target value.

Let x be an arbitrary vector and y its corresponding unknown target variable. The expected value $\hat{y}$ of variable y needs to be derived from the training set $\mathcal{L}$. Each element $\mathrm{e_i}$ of the training set is a piece of evidence concerning the value of y. The relevance of each element in $\mathcal{L}$ can be assumed to depend on the similarity between the arbitrary vector x and the input vectors x_i of $\mathrm{e_i}$. It can be assumed that a suitable discount function that depends on a distance measure $\|\cdot\|$ can measure this similarity. If x is close to a vector x_i in $\mathcal{L}$, it is assume that y is also close to y_i, which makes the element $\mathrm{e_i}$ an important piece of evidence. If x and x_i are at a great distance, it is safe to assume that y_i has a small effect on y, and provides only marginal information concerning y. The Minkowski distance will be used, it is defined as:

$$\mathrm{d}(x_\mathrm{i}, x_j) = \Big(\sum_{k=1}^{l} |x_{\mathrm{i}k} - x_{jk}|^p \Big)^{\frac{1}{p}}. \tag{3.3}$$

Where $x_{\mathrm{i}k}$ and x_{jk} are the values of vectors x_i and x_j at dimension k. When p is 1 the L1 or Manhattan distance is obtained and with p equal to 2, L2 or also known as the Euclidean distance is calculated. In the case of high dimensional spaces, the vectors become uniformly distant from each other, the ratio between the nearest and farthest vector approaches 1. This phenomenon is more present in the Euclidean than in the Manhattan distance metric [53, 54], which makes the Manhattan distance to yield better results in distance-based algorithms in the presence of a high dimensional space. EVREG could use any distance measure (e.g. Cosine similarity) to represent the similarity between two vectors. The Minkowski distance will be used for its flexibility while testing the performance for values $p \in [1, 2]$. The value of p will be chosen as the one that yields the best performance in prediction and features selection tasks. The discount function ϕ between vectors x_i and x_j using this distance measure will be defined as:

$$\phi(\mathrm{d}(x_\mathrm{i}, x_j)) = \exp\Big(-\frac{\mathrm{d}(x_\mathrm{i}, x_j)^2}{\gamma}\Big). \tag{3.4}$$

Where ϕ is a decreasing function from $\mathbb{R}$ to $[0, 1]$ that needs to fulfill $\phi(0) \in]0, 1[$ and,

$$\lim_{\mathrm{d}\to\infty} \phi(\mathrm{d}) = 0. \tag{3.5}$$

Eq. 3.4 is the well-known Radial Basis Function (RBF) that decreases monotonically with distance, commonly used as a kernel in the Support Vector Machine (SVM) classifier. Parameter γ is the radius of the function, intuitively γ defines how far the influence of a vector reaches. In Figure 3.1 eq. 3.4 can be observed for different γ values. RBF-based methods are an active area of research [55, 56]; various approaches exist to find the optimal parameters such as γ. This parameter can be learned using an optimization approach, but often the value is set manually by trial and error. The optimal γ will be found using a Grid Search approach.

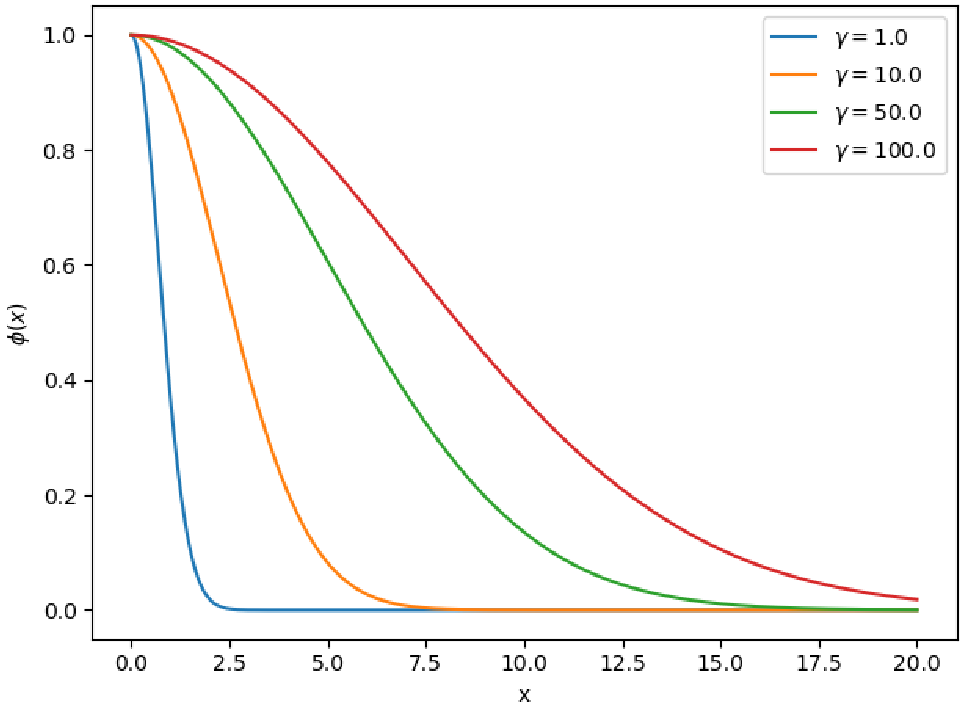


Figure 3.1: Radial Basis Function for different γ values

The discount function ϕ will represent the similarity between two vectors (higher value means higher similarity); this similarity function can be used to compute the mass (influence) of each element in $\mathcal{L}$ given an arbitrary vector x using the Dempster rule of combination getting:

$$m_{\mathrm{i}}(x) = \frac{1}{K}\phi(\mathrm{d}(x, x_{\mathrm{i}})) \prod_{k!=\mathrm{i}} (1 - \phi(\mathrm{d}(x, x_k))). \tag{3.6}$$

Where K is a normalization term defined by the DST as:

$$K = \prod_{j=1}^{N} (1 - \phi(\mathrm{d}(x, x_{\mathrm{i}}))) + \sum_{\mathrm{i}=1}^{N} [\phi(\mathrm{d}(x, x_{\mathrm{i}})) \prod_{k!=\mathrm{i}} (1 - \phi(\mathrm{d}(x, x_k)))]. \tag{3.7}$$

In eq. 3.6, ϕ determines the influence of x_{i} in x and the product determines the effect of the evidence set between this two vectors. With the previous formulas the influence of every vector in set $\mathcal{L}$ can be obtained, but one extra piece of evidence needs to be considered. In every example of set $\mathcal{L}$ the values of variable y have been observed and the EVREG takes this information into account. The variable y will be assumed to be bounded to the interval $[y_{inf}, y_{sup}]$, defining for this interval an additional mass called domain mass m^* calculated as:

$$m^*(x) = \frac{1}{K} \prod_{\mathrm{i}=1}^{N} (1 - \phi(\mathrm{d}(x, x_{\mathrm{i}}))). \tag{3.8}$$

Due to the assumption that y is bounded to interval $[y_{inf}, y_{sup}]$ a probabilistic density function $P(x)$ associated to $m_{\mathrm{i}}(x)$ and $m^*(x)$ exists. Smets et al. [57] showed that the Pignistic transformation could be used to transform the masses in EVREG to a probability function. In the particular case where output y is a real number, the Pignistic probability is defined as:

$$P(x) = \sum_{\mathrm{i}=1}^{N} m_{\mathrm{i}}(x) \cdot \delta_{\mathrm{i}}(x) + \frac{m^*}{y_{sup} - y_{inf}}. \tag{3.9}$$

Eq. 3.9 is a mixture of Dirac distributions and a continuous uniform distribution. With this probability function, the expected value of target variable y can be obtained as the Pignistic expectation [15]:

$$\widehat{y} = \sum_{\mathrm{i}=1}^{N} m_{\mathrm{i}}(x) \cdot y_{\mathrm{i}} + \frac{m^*(x) \cdot (\sup_{y \in \mathcal{L}} y + \inf_{y \in \mathcal{L}} y)}{2}. \tag{3.10}$$

Where $\hat{y}$ is the expected or predicted value of target variable y. With the Pignistic expectation the upper and lower bound for variable $\hat{y}$ can be defined as:

$$\widehat{y}^* = \sum_{\mathrm{i}=1}^{N} m_{\mathrm{i}}(x) \cdot y_j + m^*(x) \cdot \sup_{y \in \mathcal{L}} y, \tag{3.11}$$

$$\widehat{y}_* = \sum_{\mathrm{i}=1}^{N} m_{\mathrm{i}}(x) \cdot y_j + m^*(x) \cdot \inf_{y \in \mathcal{L}} y. \tag{3.12}$$

It can be observed that the interval $[\hat{y}_*, \hat{y}^*]$ contains variable $\hat{y}$. The width of this interval can be interpreted as the uncertainty of the response, which is directly associated with the mass of the domain of target variable y.

3.2 Improving Computing Time

The computation time of a prediction grows quickly with the size of the training set. For a single prediction in training set $\mathcal{L}$ of size n, the mass of each element $\mathrm{e_i}$ needs to be computed which has a vector x_{i} of dimension q. First, the discount function (ϕ) of the input vector x with every element $\mathrm{e_i}$ in the set $\mathcal{L}$ is pre-computed. This takes $\mathcal{O}(q)$ time for each element in the set, taking a total time of $\mathcal{O}(nq)$ to compute each mass of $\mathcal{L}$ additionally to the discount function. In eq. (3.6) a product sequence needs to be computed and the normalization term K. As the discount functions for the training set is pre-computed, the product sequence takes only $\mathcal{O}(log(n))$. As for the normalization term K (3.7) which takes most of the calculation time for computing eq. 3.6, the product sequence on the left side can also be computed in $\mathcal{O}(log(n))$. The sum of the right side takes time $\mathcal{O}(nlog(n))$ because of the product sequence contained in it. The domain mass also requires the normalization term K and the product sequence takes time $\mathcal{O}(nlog(n))$. Thus the time complexity for a single mass in the training set is $\mathcal{O}(nlog(n))$. Which results in a time complexity of $\mathcal{O}(n^2log(n))$ for a single prediction.

All the masses of the training set need to be calculated to compute a single prediction. The mass calculation needs to be computed n times; since a single mass is computed in time $\mathcal{O}(nlog(n))$, the n masses will take time $\mathcal{O}(n^2log(n))$. Computing a prediction with the formula shown in eq. (3.10) uses the masses of the set $\mathcal{L}$, and the mass of the domain of target variables in the training set. This calculation requires to compute the maximum and minimum values, which needs time $\mathcal{O}(n)$, so the prediction of input vector x can be computed in time $\mathcal{O}(n^2log(n))$.

The quadratic growth of each prediction makes it impossible for the EVREG model to work with large sets; therefore it is difficult to apply it for real-life problems. However, it has been suggested the use of a k-Nearest Neighbors (k-NN) approach [15] to improve computation time. The result of these study showed that a k-NN approach does not introduce a significant penalty to the algorithm performance. It is possible to create indexes for the k-NN search in $\mathcal{O}(n(q+k))$ with this approach in the following way. First, the distance between x and x_i is computed for each vector in the training set, then iterating through the distances k times selecting the smallest distance. Now, only the masses of the k nearest neighbors have to be calculated when computing a new prediction of a vector x. The masses of samples that are not in the set of the nearest neighbors of x will be assumed to be null. In particular, a flat index implementation by Johnson et al. [58] is used, for the exact nearest neighbor's search given its better execution times and memory usage compared to other existing solutions.

Given now that only the masses of the k nearest neighbors are relevant for a prediction, the mass of each sample in the training set for an input vector x is calculated as:

$$m_\text{i}(x) = \begin{cases} \frac{1}{K}\phi(\text{d}(x,x_\text{i}))\prod_{\substack{k!=\text{i} \\ x_k\in N(x)}}(1-\phi(\text{d}(x,x_h))) & \text{if } x_\text{i} \in N(x), \\ \\ 0 & \text{otherwise.} \end{cases} \tag{3.13}$$

Where $N(x)$ is the set of k nearest neighbors of vector x. Also, now the normalization term K can be computed as:

$$K = \prod_{x_\text{i}\in N(x)}(1-\phi(\text{d}(x,x_\text{i}))) + \sum_{x_\text{i}\in N(x)}[\phi(\text{d}(x,x_\text{i}))\prod_{\substack{h!=\text{i} \\ x_h\in N(x)}}(1-\phi(\text{d}(x,x_h)))]. \tag{3.14}$$

Just the k neighbors are considered as evidence for a prediction of the domain mass, so the domain evidence is only attributed to its neighbors. The calculation of the domain mass is computed as:

$$m^*(x) = \frac{1}{K}\prod_{x_\text{i}\in N(x)}(1-\phi(\text{d}(x,x_\text{i}))). \tag{3.15}$$

Since now only the masses of the k nearest neighbors are computed, the size of the training set n is changed by the number of neighbors k instead in the expression for computing the time complexity. In eq. 3.13 once we the k nearest neighbors are obtained, the normalization term K will be computed in $\mathcal{O}(qklog(k))$. The discount function will still take $\mathcal{O}(q)$ and the product sequence will have $\mathcal{O}(qklog(k)$ complexity. Only the masses of the k neighbors will be needed to compute a prediction, so only k masses of the training set will be computed thus obtaining a complexity for a single prediction of $\mathcal{O}(qk^2log(k))$. Assuming that k is always significantly less than n, the conclusion is that the complexity for a prediction ends up being reduced to $\mathcal{O}(nqk)$ because of the complexity of the k-NN search.

The following experiment was performed to demonstrate the importance of the k-NN approach for speeding up the prediction computing time. Two different settings will be tested, one for each approach: i) the first approach uses all the training set as evidence; starting with 1,000 examples and end in 30,000 with steps of size 500 in this case. ii) the k-NN approach; the examples start at 100,000 and finish in 5,000,000 examples with steps of size 100,000. The number of neighbors was fixed to 10 and the number of dimensions used was only 1. The results are shown in Figure 3.2. Each value shown in the figure corresponds to the mean execution time of 100 experiments.

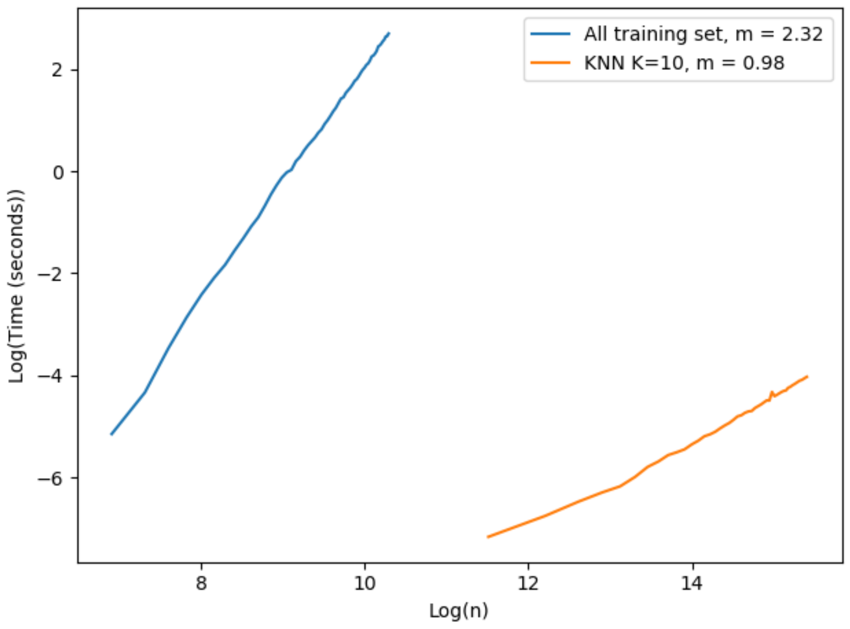


Figure 3.2: Time complexity single prediction

The plot in Figure 3.2 is in logarithmic scale. The slope of each curve was computed to observe the complexity of each approach. The x axis corresponds to the logarithmic number of samples in the training set, and the y axis is the logarithmic execution time of a single prediction in seconds. The number of vectors vary in each approach due to hardware constraints: Case i) The number of vectors that could be tested was limited due to excessive execution time and memory consumption. Case ii) the machine where the experiment ran could not register execution times properly for a low number of vectors in the training set. As expected, the time complexity of the implemented EVREG with no optimization grows dramatically faster than the k-NN approach as seen by their slopes. The complexity of using all the training set approach seems to have a larger time complexity than theoretically expected, as the slope in a complexity of $\mathcal{O}(n^2 log(n))$ is 2.12. In the k-NN approach a linearity with the size of the training set can be easily observed. Consequently, it becomes obvious that the k-NN approach significantly drops execution times and makes EVREG a viable option for real world problems.

3.3 Weighted Evidential Regression

This section proposes an improvement to the discount function used in EVREG based on ideas which has been previously introduced to enhance the well-known k-Nearest Neighbors Regressor (k-NN Regressor) [59], which is another regressor, similar to EVREG. The improved model will be called Weighted Evidential Regression (WEVREG). The aim of this improvement is to boost prediction performance and allow the model to perform feature selection tasks.

The k-NN Regressor is a simple and intuitive non-parametric regression method to estimate the output value of an unknown function for a given input. It "guesses" the output value using samples of known values as a training set, and computing the mean output value of the nearest neighbors of the input vector. All neighbors in this method are equally relevant for predicting the target variable in its original version. There is a variation of this algorithm where the importance of each neighbor depends on a distance measure between them, thus using a weighted average of the k-NN vectors in the training set. This weighted k-NN Regressor is a kind of locally weighted regression [60]. The weight of each neighbor is proportional to its proximity to the input vector. The prediction is computed with this approach by the following expression:

$$\hat{f}(x) = \frac{1}{Z} \sum_{x' \in N(x)} f(x') \, \mathrm{e}^{\frac{-\mathrm{d}(x,x')}{\beta}}, \tag{3.16}$$

where $N(x)$ is the set of k nearest neighbors of vector x, $f(x')$ is the value of the target variable of neighbor x', $\mathrm{d}(x, x')$ is a distance function between vector x and its neighbor x' (commonly the Euclidean distance), β is a parameter of the estimator and Z is a normalization function with value $Z = \sum_{x' \in N(x)} \mathrm{e}^{\frac{-\mathrm{d}(x,x')}{\beta}}$.

Similar to the EVREG, the weighted k-NN regression normally uses the Euclidean distance and a Gaussian Radial Basis discount function to assign a weight to each one of its neighbors. Further improvements have been made to the weighted k-NN Regression based on the assumption that the target variable is most accurately predicted using only the most relevant features of the neighbors which adds a Feature Selection process to the algorithm. Navot et al. [39] introduced a weighted distance function for the weighted k-NN Regressor, which improved the model performance and enabled the model to perform a feature selection task. Given a weights vector w over the features of size q the distance function induced by w is defined as:

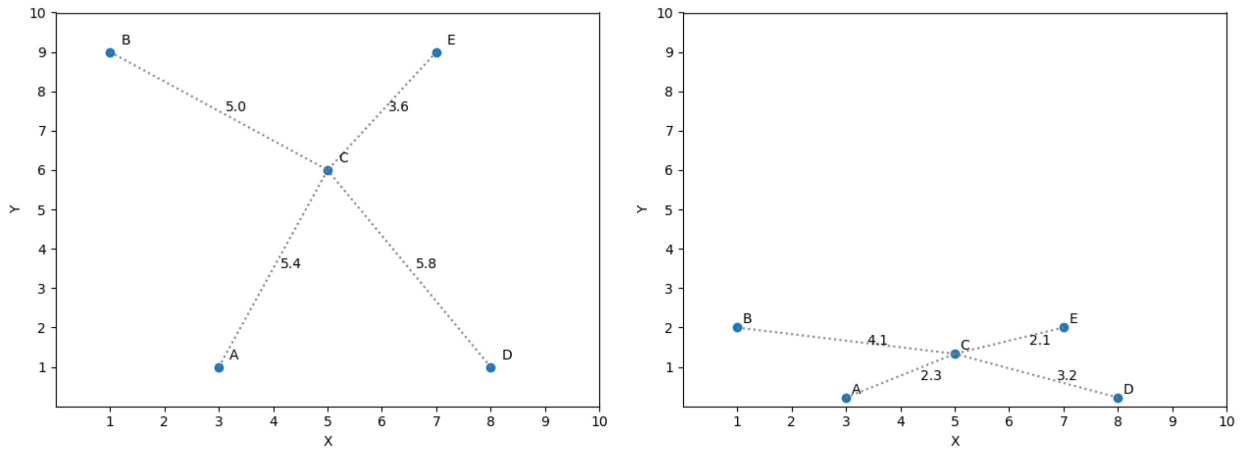
$$\mathrm{d}(x_{\mathrm{i}}, x_j) = \Big(\sum_{k=1}^{l} |(x_{\mathrm{i}k} - x_{jk}) \cdot w|^p \Big)^{\frac{1}{p}}, \tag{3.17}$$

where input vector x_{i} and x_j have the same size q as the weights vector w. Each value of vector w represents the importance for each dimension in computing the distance between two vectors, i.e. the amount which a feature contributes to the distance between these two vectors. In EVREG, eq. (3.3) is assumed that every dimension contributes the same to the distance between two vectors, and consequently, to the discount function 3.4. However, this is not always the case; for example a feature in the input that has no impact in the target variable that is been predicted thus it contributes nothing to the similarity between those vectors. For instance, on a medical sense, a patient age could have a great impact on an individual health expenditures so to predict their expenses we want to be closer to patients in the same age group, but maybe another variable such as eyes color does not influence their expenses in any way, so it would not be desired to include a patients with their same eyes color just for this fact.

The distance function in eq. (3.3) will be changed with the one that uses weights for each dimension as presented in eq. (3.17) in order to improve the prediction performance of the EVREG model and to gain further understanding about the data.

The distance measure described in eq. (3.17), will also compute the k-NN of each input vector x. The k-NN search implementation utilized so far does not have a feature to use a custom distance measure to compute a k-NN search, so indexes with a transform training set space (applying the weights vector) will have to be created, and then transform the input vector x that will be predicted. With this operation the same distance measure as described in eq. (3.17) will be obtained.

A simple example is presented to ease the understanding of the chain of thoughts behind this idea. Let assume the sample set is of size 5 in a two dimensional space and the third dimension needs to be predicted. The input vectors (two dimensions) are shown in Figure 3.3a. The optimization discussed in Section 3.2 is used in this example, taking into account only the three closest neighbors to predict the target variables. If the output value for vector C is predicted using only its three closest neighbors, then A, B and E should be considered, as is shown in 3.3a by the distance between C and all the vectors. Nevertheless, what if its found that dimension y is not as significant as x for predicting the target dimension? This observation means a variation in y does not affect in the same magnitude the prediction as a difference of the same size in x. The space to reflect the variation in magnitude is scaled, getting Figure 3.3b.



(a) Example input vectors. (b) Transformed input vectors.

Figure 3.3: Feature transformation.

If the three closest neighbors for vector C are computed then we get vector A, E and D, replacing vector B, thus reflecting the importance of feature x in the similarity function.

For the weight of a feature to represent the importance in an input vector, each one of the feature domains has to be of the same size. If the size of the domains is not equal, a bigger weight of a dimension could be reached, only to compensate on the size of the domain. Is recommended to normalize the input features (all values must be centered in 0 between -1 and 1) so that the weights are comparable between features.

The optimal weight vector can be found using an optimization approach such as gradient descent. In the next section the process of finding the optimal weights for a known training set will be described.

3.4 Weight Learning

This section will present the process of finding the optimal weights vector in a known training set. Gradient descent algorithm will be used to find a vector w that minimizes the error of the proposed model.

The accuracy of a regressor is commonly measured by computing the difference between the predicted value $\hat{Y}$ and the actual value of the target variable Y. The estimation error is expressed by a loss function L(Y, $\hat{Y}$), $L : \mathrm{IR} \times \mathrm{IR} \rightarrow \mathrm{IR}$. Likewise in a Supervised Learning problem, the goal is to find a vector w that induces the smallest error in L using the training set $\mathcal{L}$ and a small generalization error in a validation set at the same time. A gradient-based algorithm such as gradient descent with an Adam optimizer will be used to get the optimal value for w, because of its clear mathematical justification for reaching optimum values [61].

The gradient descent algorithm is an iterative algorithm that optimizes variable values in order to find a function minimal value. The Mean Squared Error (MSE) is used as error function [62]. The loss function L induced by w will be defined as:

$$L_w(Y, \widehat{Y}) = \frac{1}{n} \sum_{\mathrm{i}=1}^{n} (y_\mathrm{i} - \widehat{y_\mathrm{i}})^2, \tag{3.18}$$

where n is the number of samples in the training set, Y is a vector of size n with the actual values for the target variable of each sample in the training set, and $\hat{Y}$ is the predicted vector for each one of the samples in the training set. y_i and $\hat{y}_\mathrm{i}$ are elements of Y and $\hat{Y}$ respectively.

After defining the function error L which is induced by the weights vector w, the goal is to find a vector w that yields the smallest error possible for the training set given. The estimator $\hat{w}$ for vector w is obtained by minimizing this criterion:

$$\widehat{w} = argmin\ L_w(Y, \widehat{Y}). \tag{3.19}$$

Since the loss function L_w is continuous and differentiable, gradient descent algorithm can be used to find $\hat{w}$. The gradient of L_w needs to be computed in every iteration of the algorithm in order to update the weights w, then w is updated by taking a step proportional to the negative of the gradient. So a new weight vector in iteration $\mathrm{i}+1$ is calculated as:

$$w_{\mathrm{i}+1} = w_\mathrm{i} - \alpha \cdot \frac{\partial L_w}{\partial w}. \tag{3.20}$$

Where α is the proportion of the step used also known as learning rate. The partial derivation of L_w is needed for the gradient computation. This derivation is calculated as:

$$\frac{\partial L_w}{\partial w} = \frac{2}{n} \sum_{\mathrm{i}=1}^{n} \frac{\partial (y_\mathrm{i} - \widehat{y_\mathrm{i}})}{\partial w} = \frac{2}{n} \sum_{\mathrm{i}=1}^{n} \frac{\partial (y_\mathrm{i} - \widehat{y_\mathrm{i}})}{\partial \widehat{y_\mathrm{i}}} \cdot \frac{\partial \widehat{y_\mathrm{i}}}{\partial w}. \tag{3.21}$$

The partial derivative of the predicted target variable $\hat{Y}$ need to be calculated with respect to w in order to solve eq. (3.21). Therefore, the calculation of the derivative of eq. (3.10) gives:

$$\frac{\partial \widehat{y_\mathrm{i}}}{\partial w} = \sum_{j=1}^{N} \frac{\partial m_j(x_\mathrm{i})}{\partial w} \cdot y_j + \frac{\partial m^*(x_\mathrm{i})}{\partial w} \cdot \frac{(\sup_{y \in \mathcal{L}} y + \inf_{y \in \mathcal{L}} y)}{2}. \tag{3.22}$$

Where the derivative of a single mass in set $\mathcal{L}$ is calculated as,

$$\frac{\partial m_j(x_\mathrm{i})}{\partial w} = \frac{K \frac{\partial \phi(x_\mathrm{i}, x_j)}{\partial w} - \frac{\partial K}{\partial w} \cdot \phi(x_\mathrm{i}, x_j)}{K^2} \cdot \prod_{k!=\mathrm{i}} (1 - \phi(\mathrm{d}(x, x_k))) + \frac{\phi(x_\mathrm{i}, x_j)}{K} \cdot \frac{\partial (\prod_{k!=\mathrm{i}} (1 - \phi(\mathrm{d}(x, x_k))))}{\partial w}. \tag{3.23}$$

With the derivatives of discount function ϕ and the normalization term K as,

$$\frac{\partial\phi(x_{\mathrm{i}}, x_j)}{\partial w} = -\frac{2}{\gamma}\exp(\frac{-\mathrm{d}(x_{\mathrm{i}}, x_j)^2}{\gamma}) \cdot \mathrm{d}(x_{\mathrm{i}}, x_j) \cdot \frac{\partial\mathrm{d}(x_{\mathrm{i}}, x_j)}{\partial w}, \tag{3.24}$$

$$\frac{\partial K}{\partial w} = \frac{\partial(\prod_{j=1}^{N}(1-\phi(\mathrm{d}(x, x_{\mathrm{i}}))))}{\partial w} + \sum_{\mathrm{i}=1}^{N} \frac{\partial[\phi(\mathrm{d}(x, x_{\mathrm{i}}))\prod_{k!=\mathrm{i}}(1-\phi(\mathrm{d}(x, x_k)))]}{\partial w}. \tag{3.25}$$

The derivative of the product sequence is calculated as,

$$\frac{\partial(\prod_{k!=\mathrm{i}}(1-\phi(\mathrm{d}(x_{\mathrm{i}}, x_k)))}{\partial w} = -\sum_{\substack{h=1\\h!=\mathrm{i}}}^{n} \frac{\partial(\phi(x_{\mathrm{i}}, x_h))}{\partial w} \cdot \prod_{\substack{k=1\\k!=h}} (1-\phi(x_{\mathrm{i}}, x_k)). \tag{3.26}$$

Finally, the derivative of the mass assigned to the domain can be calculated as,

$$\frac{\partial m^*(x_{\mathrm{i}})}{\partial w} = \frac{K\frac{\partial(\prod_{\mathrm{i}=1}^{N}(1-\phi(\mathrm{d}(x,x_{\mathrm{i}})))}{\partial w} - \frac{\partial K}{\partial w}\prod_{\mathrm{i}=1}^{N}(1-\phi(\mathrm{d}(x, x_{\mathrm{i}})))}{K^2}. \tag{3.27}$$

Then the weight learning algorithm can be computed predicting the target variable y_{i} of every example e_{i} in the training set $\mathcal{L}$ using a leave-one-out approach. When the output value of a sample e_{i} is predicted, this sample is left out from set $\mathcal{L}$ and all the other $n-1$ samples are used for which the actual output value is known in the set as evidence. This operation will be performed multiple times through all the training set, a single pass through all the training set will be called an epoch. The algorithm will perform a fixed number of epochs. At the end of each epoch the weights will be updated by the calculated gradient multiplied by a learning rate factor α (eq. 3.20) and the weight vector will be stored only if a local minimum is found. Finally, the method will return the weight vector $\hat{w}$ which minimizes the loss function L, as is shown in Algorithm 3.

The introduction of the weight learning process makes the EVREG model not only better predict the output of a given variable, but the method also also gains the ability to perform feature selection tasks. The main advantage of EVREG is its transparency. This means we can easily track any prediction made by the model; the contribution (mass) of each piece of evidence in the training set $\mathcal{L}$ is known. The model can now present a ranking of features according to their importance (weight), i.e., the contribution of a feature to each mass computed, thus helping the end user get a better understanding of her/his data. This improved EVREG will be named Weighted Evidential Regression (WEVREG).

Algorithm 3 Weight learning

```
 1: function WEIGHT LEARNING($X, Y, \alpha, NumEpochs$)
 2:     $w \leftarrow [1, 1, \ldots, 1]$
 3:     $minLoss \leftarrow \infty$
 4:     $\hat{w} \leftarrow w$
 5:     for $epoch \leftarrow (1, NumEpochs)$ do
 6:         $\hat{Y} \leftarrow [\,]$
 7:         for $x_i \in X$ do
 8:             $S \leftarrow Remove(x_i, X)$                    ▷ Remove $x_i$ from training set
 9:             $\hat{y}_i \leftarrow EVREG(x_i, S, w)$          ▷ Predict example $e_i$ with training set S
10:             $Insert(\hat{y}_i, \hat{Y})$                     ▷ Insert $\hat{y}_i$ in $\hat{Y}$
11:         end for
12:         $loss \leftarrow L(Y, \hat{Y})$                      ▷ Compute loss
13:         $gradient \leftarrow CalculateGradient(Y, \hat{Y})$  ▷ Compute gradient
14:         $w \leftarrow w + \alpha * gradient$                ▷ Update weights vector
15:         if $minLoss > loss$ then                           ▷ Save best weight vector
16:             $minLoss \leftarrow loss$
17:             $\hat{w} \leftarrow w$
18:         end if
19:     end for
20:     return $\hat{w}$
21: end function
```

Chapter 4

Model Evaluation

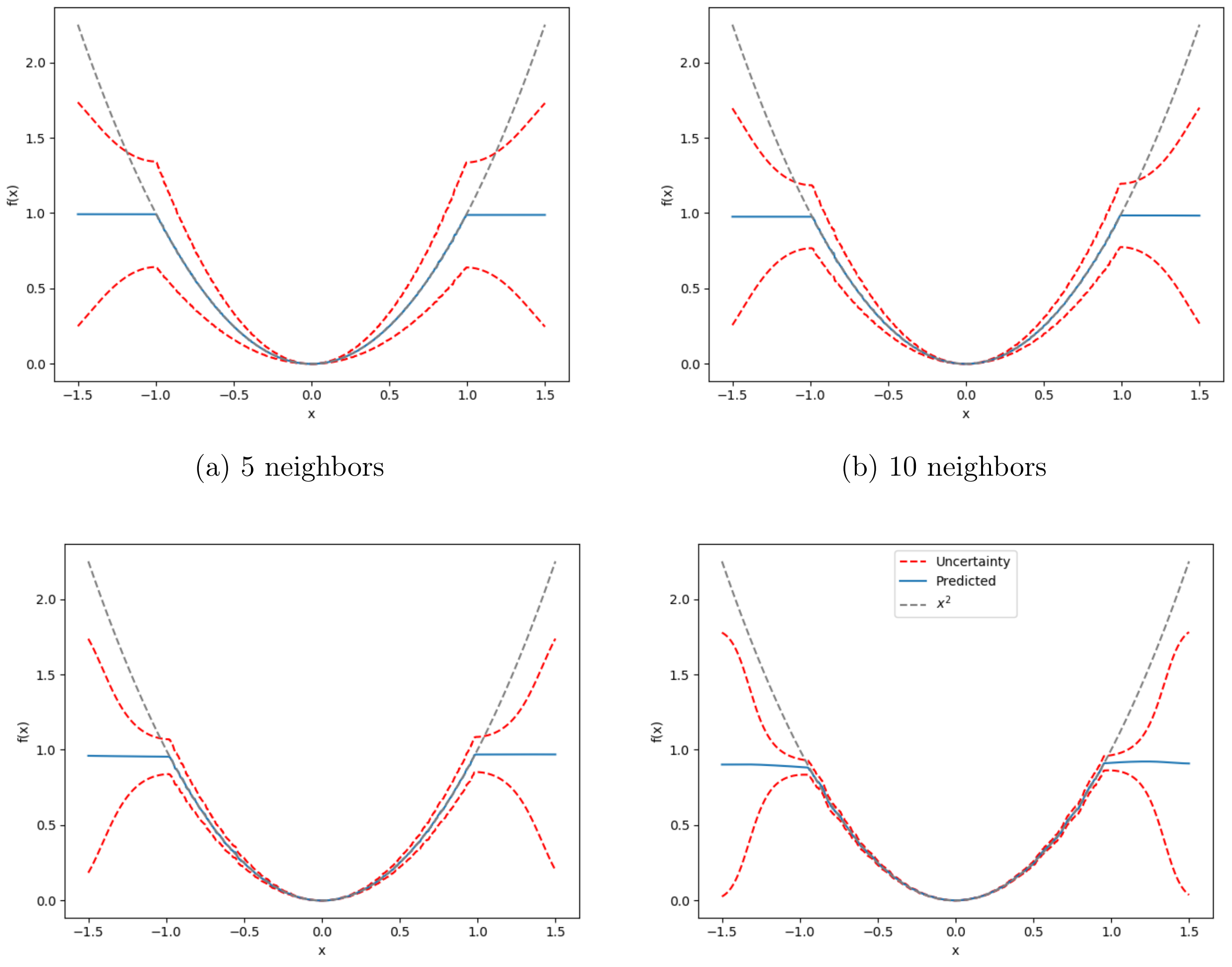
In this section, the performance of the model in prediction and feature selection tasks will be tested. Well known regression and embedded methods will be used as benchmarks. Quantitative performance measures will be introduced to objectively measure the performances of our model against the benchmark methods. For FS tasks, a measure proposed by Bolón-Canedo et al. [52], called *index of success* will be used. This index will be applied in synthetic data experiments, because the relevant features are known *a priori*. Commonly used error measures for prediction performance such as Relative Mean Squared Error (RMSE) will be used in well known regression problems.

4.1 Synthetic data

In this section synthetic data will be used to demonstrate the ability of the proposed method to solve prediction and FS tasks. First, two small experiments will be performed to illustrate its prediction ability. Then, a small example of a feature selection problem will be performed, demonstrating the learning process of the features importance. Finally, a quantitative measure for FS performance will be introduced and tested in four different synthetic datasets.

Two single dimension functions will be used in order to illustrate how the method behaves. A parameter regarding the number of neighbors is used to demonstrate the method's sensitivity over the prediction and uncertainty of the response according to this particular parameter. The functions that will be predicted are $f(x) = x^2$ and $f(x) = sin(2\pi x + \frac{\pi}{2})$. Using $1,000$ points as a sample set. The points will be selected at random from a uniform distribution over the interval $[-1, 1]$. In each function, the sample set will be split randomly using 950 points as the training set, and only 50 points as validation. The method will receive the training set $\mathcal{L}$ with the input X of 950 points and output $f(x)$ of these points. In Figure 4.1 the predicted values and uncertainty boundaries for function $f(x) = x^2$ are shown with the number of neighbors under each figure.

In Figure 4.1, the dashed red lines represent the upper and lower expected values for the predicted function $f(x)$. These expected values represent the uncertainty of the prediction.



(a) 5 neighbors

(b) 10 neighbors

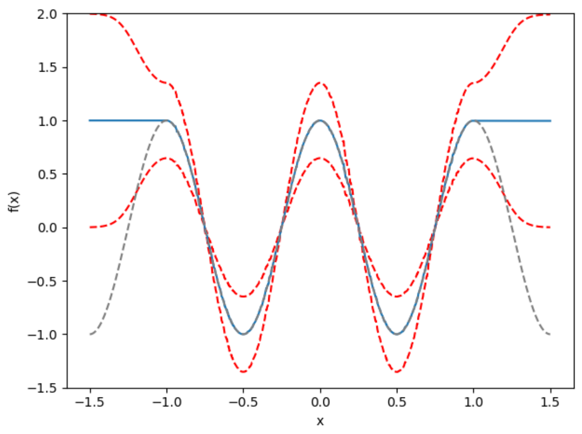
(c) 20 neighbors

(d) 50 neighbors

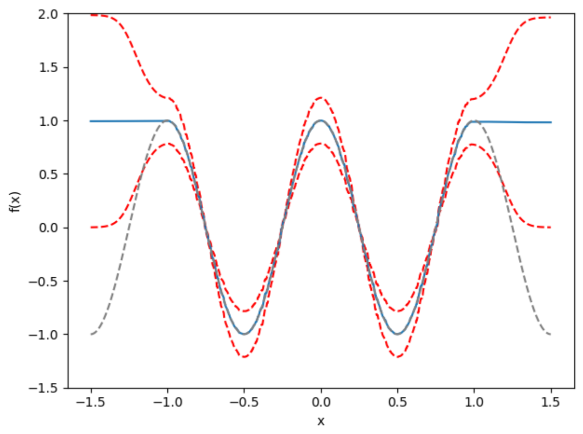
Figure 4.1: Prediction for $f(x) = x^2$

As it is shown, the prediction between interval $[-1, 1]$ follows the desired function. When the method tries to extrapolate the values of the function where there are no samples (outside interval $[-1, 1]$), the uncertainty of the response increases rapidly with the distance. This results in a constant prediction outside the defined interval. Also, when neighbors increase, the uncertainty of the response decreases as the uncertainty curves get closer to the predicted curve. For function $f(x) = sin(2\pi x_1 + \frac{\pi}{2})$ the same experiment was performed. The sample set is also bounded to $[-1, 1]$. The train and validation sate are also split as the previous function. The results for the predicted function are presented in Figure 4.2.

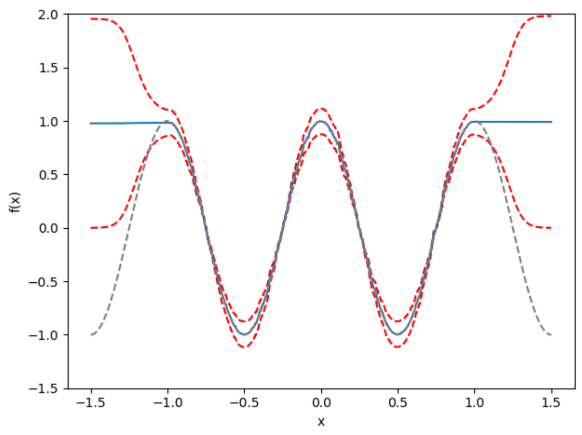
As shown in Figure 4.2, the prediction fits perfectly where the sample set is defined. As expected, when the method extrapolates, uncertainty grows rapidly with the distance. For this function, the predictor has a higher uncertainty in local maxima and minima due to the low number of points in these areas. Again when the number of neighbors grow, uncertainty decreases making the red curves to get closer to the predicted function. When the mass of the target variable domain reaches its maximum, the points leave the bounded interval of the function and the uncertainty remains constant.



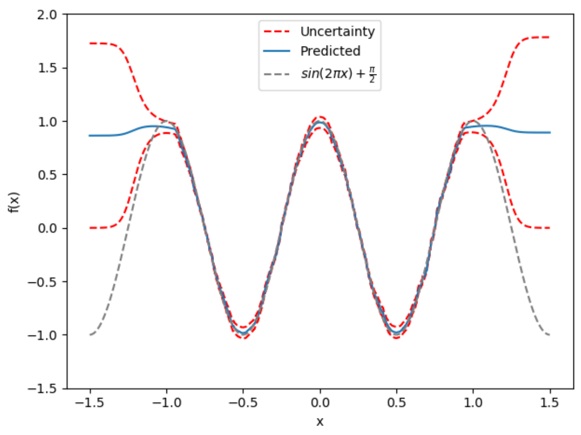
(a) 5 neighbors



(b) 10 neighbors



(c) 20 neighbors



(d) 50 neighbors

Figure 4.2: Prediction for $sin(2\pi x) + \frac{\pi}{2}$

Another small experiment will be presented to test the FS ability of the method. Using a 5-dimensional set of vectors $X = (x_0, x_1, x_2, x_3, x_4)$ with 100 samples. Each vector will be randomly selected from a uniform distribution in a $[0, 1]^5$ cube. $f(X) = x_0$ will be used as the target function, using all vectors as a training set. As explained in Section 3.4, the weight learning process is computed through a number of epochs, which is an iteration over all samples available. The value of every weight starts at 1 and in each iteration, the weight values $(w_0, \ldots, w_4)$ are updated. Before any epoch is computed, the error obtained from the model is the same that would be obtained from EVREG. The MSE (eq. 3.18) of the method in each epoch can be observed in Figure 4.3.

As shown in Figure 4.3, the error decreases rapidly. When the method stops learning (the error stops decreasing), it is expected to obtain the most important feature x_0, since the first dimension of the input gives its value to $f(x)$. The learning process of the weight vector can be seen in Figure 4.4, where the importance of every feature in each epoch is shown.The proposed method is able to detect the main feature, since every feature starts with an initial value of 1. Quickly, all the irrelevant features get close to 0, while the value of x_0 increases, obtaining x_0 as the only important feature.

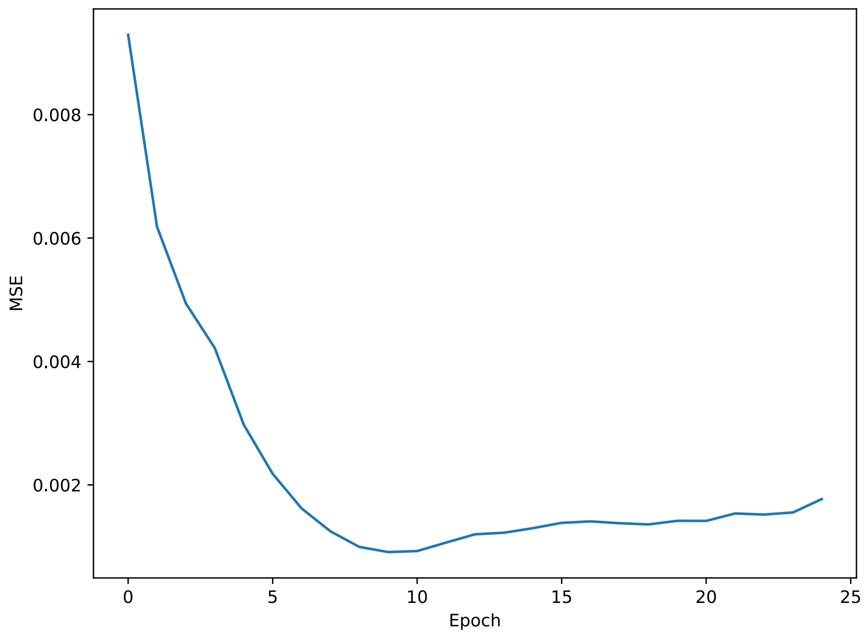


Figure 4.3: Learning phase $f(x) = x_0$

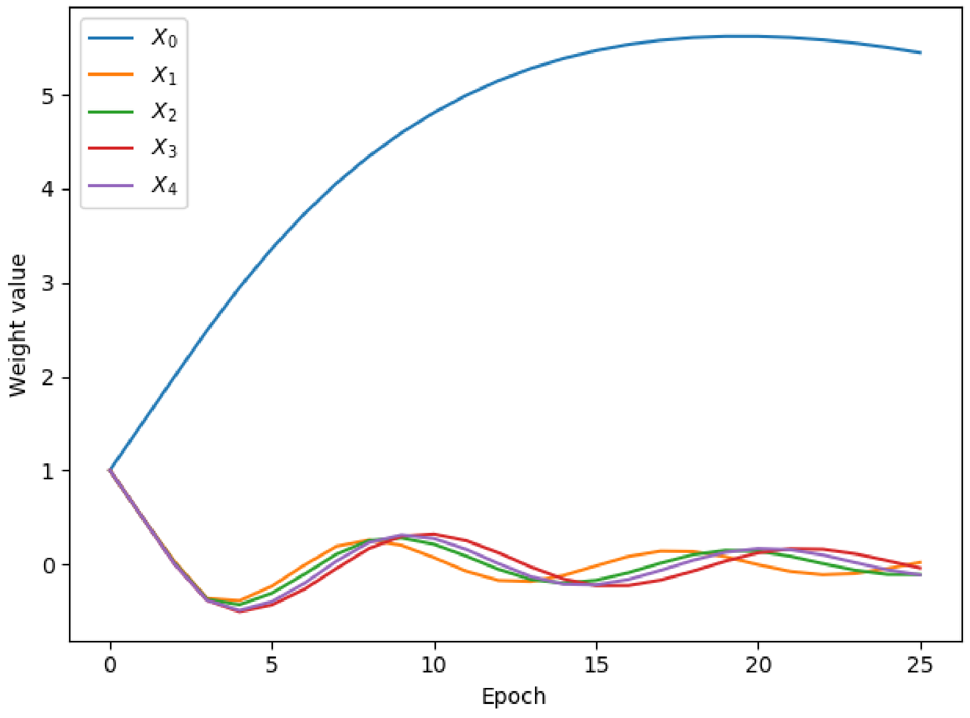


Figure 4.4: Weight learning process $f(x) = x_0$

Next, to objectively measure the FS performance, the *index of success* (SUC) introduced by Bolón-Canedo et al. [52] will be used. SUC is a scoring measure to fairly compare the effectiveness of different feature selection methods. This measure rewards the inclusion of relevant features and penalizes the selection of irrelevant ones. SUC is defined as:

$$SUC = \frac{R_s}{R_t} - \alpha\frac{I_s}{I_t}. \tag{4.1}$$

Where R_s is the number of relevant features selected, R_t the total number of relevant features, I_s the number of irrelevant features selected and I_t the total number of irrelevant features. The term α is a term to penalize the miss of a relevant feature over the include of an irrelevant one. Term α is defined as $\alpha = min\{\frac{1}{2}, \frac{R_t}{I_t}\}$. The higher the value of this expression, the better the method. SUC is bounded to interval $[-\frac{1}{2}, 1]$.

Synthetic datasets are used, so that the important features can be known. With this approach the performances of the evaluated methods can be precisely measured. Four different functions will be used in a $[0, 1]^{50}$ cube space. In every function, 4 sample sets of randomly distributed vectors from a normal distribution over $[0, 1[$ will be used. The size of the sets range from 25 to 100 in steps of 25. In each one of the sets, all vectors will be used as the training set to obtain the feature importance of the input.

There have been studies of FS in regression tasks where synthetic data is used. Navot et al. [39] used four functions with different levels of complexity to demonstrate the performance of a k-NN based FS algorithm in a regression setting. Other works [34, 63] have used the Friedman regression problem [64]. In this work, two functions from Navot et al. [39] work will be used alongside the logical XOR and Friedman function. Two of them have simple dependencies where two or more variables need to be considered independently. The remaining functions are complex, where two variables need to be considered at the same time to detect their importance. Each function is listed down below:

$$f(x) = sin(2\pi x_0) + x_1, \tag{4.2}$$

$$f(x) = 10sin(\pi x_0 x_1) + 20(x_2 - 0.5)^2 + 10x_3 + 5x_4, \tag{4.3}$$

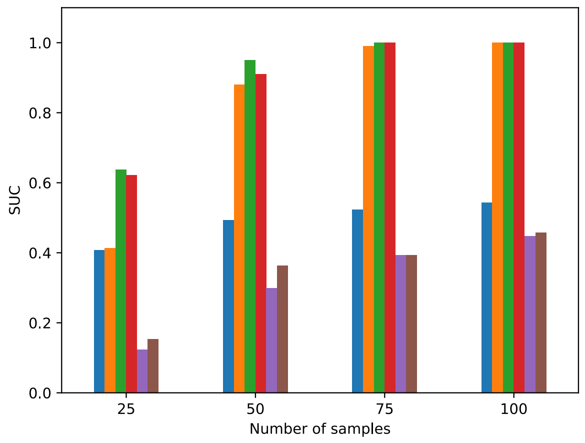
$$f(x) = (x_0 \vee x_1) \wedge \neg(x_0 \wedge x_1), \tag{4.4}$$

$$f(x) = sin(2\pi x_0) \cdot sin(2\pi x_1). \tag{4.5}$$

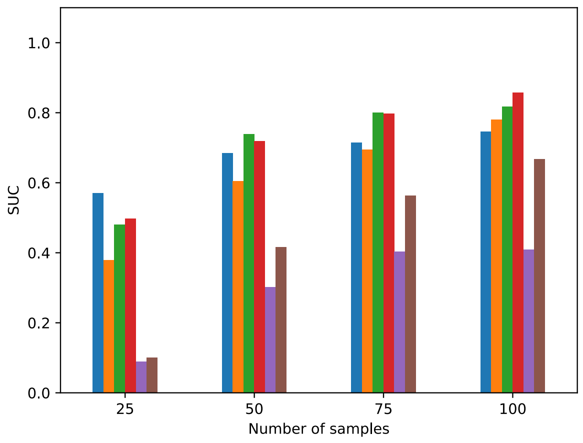
The first function in Eq. 4.2, is a non-monotonic function with a simple dependency of two features. The next function (Eq. 4.3) is the Friedman function, which has 5 relevant features. Eq. 4.4, is the logical XOR function that has two binary relevant features with a complicated dependency which requires the consideration of two features simultaneously. The last function (Eq. 4.5) is a XOR-like function with a continuous output.

The proposed regression method (WEVREG) could use any distance function to compute the similarity between vectors. For generalization the Minkowski distance (eq. 3.3) was selected for this work. In particular, the distance function used will be the L2 or Euclidean distance, which is a Mikowski distance with $p = 2$. Other values were tested but the Euclidean distance was the one that yields the best performance. For benchmarking, 5 other methods will be used. The first is the Pearson‘s correlation coefficient (CORR) [65] that is used as baseline, as it is one of the most popular correlation measures. Furthermore, other tree based algorithms will be used as comparison, such as Regression Tree (RT), Random Forest (RF) and Gradient Boosting (GB). The remaining method will be a k-NN based algorithm similar to the one WEVREG uses in its learning phase to rank every feature in a set. This is the Weighted k-Nearest Neighbors (WkNN) [34]. All tested methods are embedded ones, except for the correlation coefficient used as a baseline.

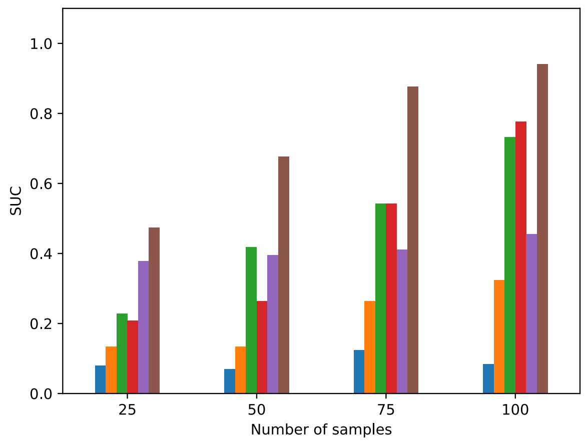
To objectively evaluate the SUC index between methods,the top 10% of the ranked features by every method will be selected. The configuration used for the methods is shown in Appendix A.1.1. The performance of each method will be the mean of a 100 executions, in each sample size. The dimensions of the input vector will be randomly permuted after calculating the target variable $f(x)$ in each experiment. So that if a method ranks features in the same order that they were given would get a low SUC value (close to 0). The results are shown in Figure 4.5.



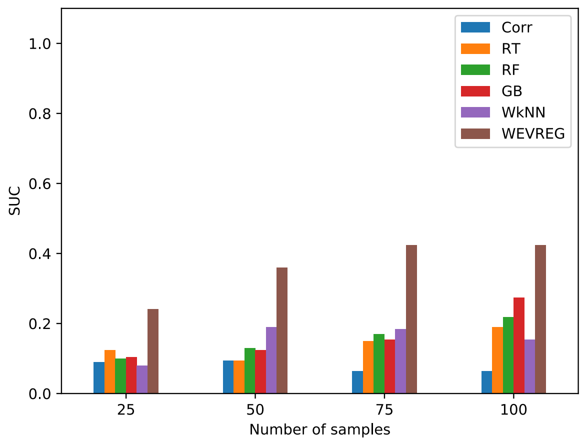
(a) $f(x) = sin(2\pi x_1) + x_2$



(b) Friedman



(c) Logical XOR



(d) $f(x) = sin(2\pi x_1) \cdot sin(2\pi x_2)$

Figure 4.5: Success index with different sample size for 4 different functions

Figure 4.5 shows the index of success achieved by each of the seven models for 4 different functions. As it is shown by Figure 4.5a, the WEVREG model performs significantly better than WkNN, the other k-NN based algorithm, and it obtains a better performance than the correlation index when the sample size grows. We can also conclude from the figure that all the tree based algorithms reach a consistent SUC, the k-NN based methods do not reach the same performance and even get worst results than CORR. It seems that for this types of problem these methods require bigger sample sizes. The Friedman function in Figure 4.5b has a simple relation between five features, but WkNN was unable to detect the importance of these features most of the time. The correlation index obtained now its best performance even surpassing WEVREG. For the XOR function in Figure 4.5c, WEVREG outperforms every method tested, requiring a smaller set for the detection of the features compared to the tree based algorithms. In the last function in Figure 4.5d, all algorithms have problems identifying the correct features but the proposed method is the one that is able to obtain the best performance. This demonstrate the ability of WEVREG to detect complex relationships.

4.2 Well-known datasets

Well known regression datasets will be used in this section, including the Boston housing and the red wine quality datasets. The aim of this section is to compare the prediction performance of the proposed method with other well known regression algorithms. We will also test its FS performance, like in the previous section.

The feature importance in the data is not known in real world problems *a priori*. So the prediction performance using a subset of features selected by the FS methods is used instead in order to test FS performance in these sets. The best FS methods will yield the best prediction performance of all FS algorithms evaluated with this approach.

The performance of WEVREG will be tested in 4 different regression problems, against other well known regression methods. The datasets that will be used are shown in Table 4.1.

Table 4.1: Datasets details

Name	**# Instances**	**# Features**
Boston housing	506	14
Medical expenses	1,338	7
Red wine quality	1,599	12
Abalone	4,177	9

Four measures will be used to assess the prediction performance of each method. The first one is the Root Mean Squared Error (RMSE), which is the root of the average squared difference between the estimated values and the actual value.

$$RMSE(Y,\widehat{Y}) = \sqrt{\frac{\sum_{\mathrm{i}=1}^{n}(y_\mathrm{i} - \widehat{y}_\mathrm{i})^2}{n}}. \tag{4.6}$$

Where Y is the vector of the true target variables and $\hat{Y}$ is the vector of its predicted values, both are vectors of size n. The effect of each difference is proportional to the size of the squared error; thus larger errors have a disproportionately large effect on RMSE. Consequently, it is sensitive to outliers.

Another measure will be the Mean Absolute Error (MAE), which computes the average absolute difference between the predicted values $\hat{y}$ and the real value y, as shown in eq. 4.7.

$$MAE(Y, \hat{Y}) = \frac{1}{n} \sum_{\mathrm{i}=1}^{n} |\hat{y}_{\mathrm{i}} - y_{\mathrm{i}}|. \tag{4.7}$$

Where $\hat{y}$ and y are vectors of size n. However, the MAE is useful to compare algorithms using the same data set; it is not useful when different data for each method is used, so the Mean Absolute Percentage Error (MAPE) will be used instead. MAPE is a modified absolute error where the difference between the predicted variable $\hat{y}$ and the real value y is divided by value y, computed as:

$$MAPE(Y, \hat{Y}) = \frac{1}{n} \sum_{\mathrm{i}=1}^{n} \frac{|\hat{y}_{\mathrm{i}} - y_{\mathrm{i}}|}{y_{\mathrm{i}}}, \tag{4.8}$$

where again n is the size of vectors Y and $\hat{Y}$. One of the disadvantages of this latter error measure is that it does not support real values that are zero. Fortunately, that is not the case in any of the used datasets.

The last measure used in this thesis will be the R^2 score, which is the Pearson correlation between the predicted and actual values; it represents how close the predicted values are to the real curve. This measure is calculated as:

$$R^2(Y, \hat{Y}) = 1 - \frac{\sum_{\mathrm{i}=1}^{n} (y_{\mathrm{i}} - \hat{y}_{\mathrm{i}})^2}{\sum_{\mathrm{i}=1}^{n} (y_{\mathrm{i}} - \bar{m})^2}. \tag{4.9}$$

Where $\bar{m}$ is the mean of variable Y defined as:

$$\bar{m} = \frac{1}{n} \sum_{\mathrm{i}=1}^{n} y_{\mathrm{i}}. \tag{4.10}$$

The prediction performance of the proposed method will be tested against the performance of other 5 embedded regression methods using each one of these error measures. For WEVREG, just the Euclidean distance will be used as we did in the previous section. The aim of this section is to show that the proposed method is able to obtain comparable results to other regression methods. The goal is not to obtain the best performance in every single problem but to obtain consistent results though out each problem so that methods can be compared in the same setting. For simplicity, each method will have a unique configuration that is used in every set. The configuration of each method will be obtained using a grid search approach. Grid search is an exhaustive search method where a set of values is used in each one of the hyper-parameters. Every combination of parameters that are possible with these sets is tested. Finally, the configuration that yields the best performance is selected. This approach will be used because of the small size of these sets. The parameters are found based on the best average performance of each method in all sets with this approach. For the k-NN based algorithms, the number of neighbors used for each one was selected based on the best performance of the proposed method. This was done to show the differences among all these methods. The configuration of each method is shown in Appendix A.1.2.

We will show the prediction performance for each method using all the features available for each dataset. Then, a variable subset of features is selected based on the features importance obtained from the FS methods. Finally, the feature importance of each is shown. Each feature in the input vector was normalized around 0 with a standard deviation of 1 for every experiment . The training and validation sets will be split randomly with 80% for the training and the remaining used for validation. Every shown performance is the mean of 100 executions.

4.2.1 Boston housing

The Boston housing dataset [66] is data collected by the U.S Census Service concerning housing in the area of the city of Boston in the mid-1970s. In this set, the price of houses is predicted using social and ecological information, for example per capita crime rate and nitric oxide concentration. The detailed feature vector is shown in Table 4.2.

Table 4.2: Boston housing feature vector

Feature	**Description**
CRIM	Per capita crime rate by town
ZN	Proportion of residential land zoned for lots over 25,000 sq.ft
INDUS	Proportion of non-retail business acres per town
CHAS	Charles River dummy variable (1 if tract bounds river; 0 otherwise)
NOX	Nitric oxides concentration (parts per 10 million)
RM	Average number of rooms per dwelling
AGE	Proportion of owner-occupied units built prior to 1940
DIS	Weighted distances to five Boston employment centres
RAD	Index of accessibility to radial highways
TAX	Full-value property-tax rate per $10,000
PTRATIO	Pupil-teacher ratio by town
BLACK	$1000(B_k - 0.63)^2$ where B_k is the proportion of african americans by town
LSTAT	% lower status of the population
MEDV	Median value of owner-occupied homes in $1000's (target variable)

The prediction performance of each method is shown in Figure 4.3. The input is normalized centered in 0 with a standard deviation of 1 in each experiment. The prediction is made using **MEDV** as target variable.

Table 4.3 shows that GB obtains the best performance closely followed by RF. WEVREG obtains results that place it at the top half of the table with similar results to WkNN. Next, the prediction performance of WEVREG using different subsets of features selected by the FS

Table 4.3: Boston housing prediction performance

Method	**RMSE**	**MAE**	**MAPE**	R^2
GB	3.22 ± 1.01	1.99 ± 0.32	0.10 ± 0.02	0.88 ± 0.07
RF	3.75 ± 0.89	2.57 ± 0.38	0.13 ± 0.02	0.84 ± 0.07
WkNN	4.57 ± 0.06	2.75 ± 0.18	0.13 ± 0.02	0.77 ± 0.00
WEVREG	4.73 ± 0.17	3.12 ± 0.14	0.15 ± 0.00	0.75 ± 0.01
ANN	4.76 ± 0.06	3.12 ± 0.17	0.16 ± 0.02	0.75 ± 0.00
RT	4.90 ± 1.20	2.92 ± 0.38	0.15 ± 0.02	0.73 ± 0.13
SVR	4.98 ± 0.03	3.38 ± 0.07	0.17 ± 0.01	0.73 ± 0.01
KNN	5.59 ± 0.18	3.43 ± 0.02	0.15 ± 0.00	0.66 ± 0.03

algorithms is shown to test the FS performance. Figure 4.6 shows the prediction performance of WEVREG with a different number of features selected by the FS methods.

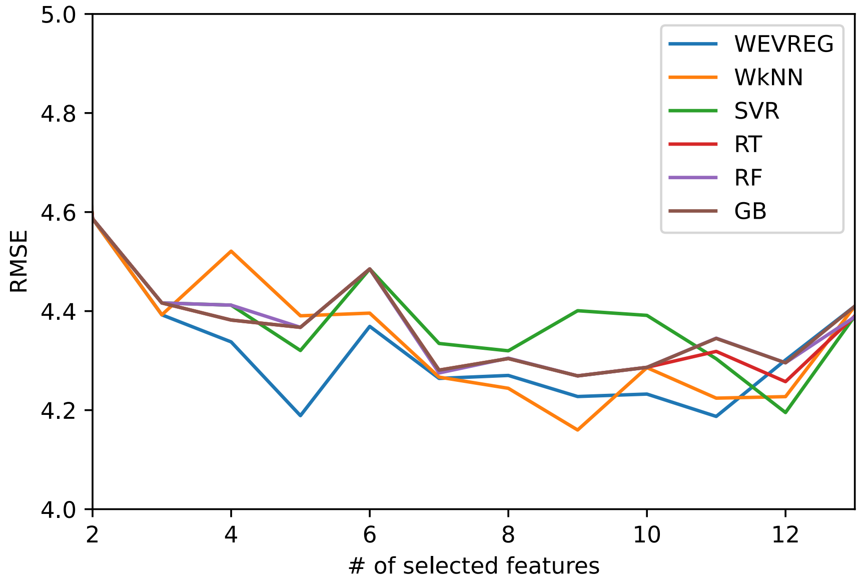


Figure 4.6: Boston housing feature selection performance

Figure 4.6 shows that all FS methods achieve similar performance. The k-NN based methods obtain the best performance. Overall, WEVREG obtains the best performance as it is one of the lowest curves in the figure, and requiring the least number of features to reach its best performance. Its best performance occurs by using the top 5 and 11 features selected. The importance given by each method is shown in Figure 4.7 as a heatmap. Each number represents the position of each feature (1 is the most important).

As shown in Figure 4.7, there is a strong agreement across the methods that the average number of rooms (RM) and the percentage of lower status (LSTAT) features are a good indicator for the price of a house in this set. WEVREG gives a higher importance to the proportion of residential land and non-retail business per town compared to the other methods even though they are overshadowed by the two main components.

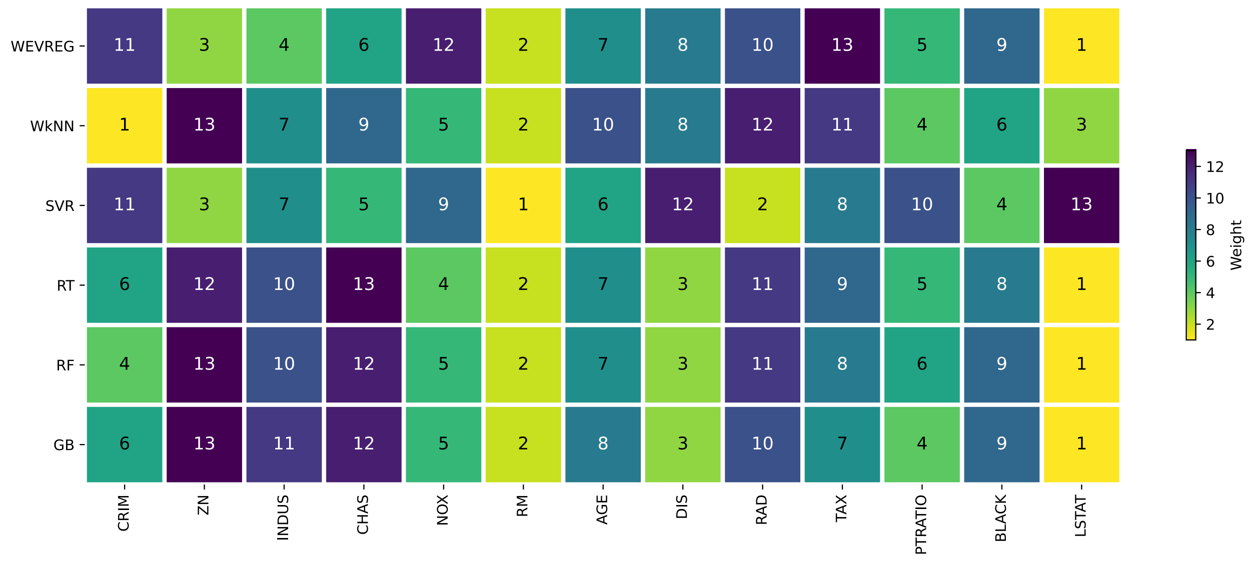


Figure 4.7: Boston housing features importance

4.2.2 Medical expenses

The medical expenses dataset [67], is a simulated dataset that contains medical expenses for patients in the United States. This data was created using demographic statistics from the U.S. Census Bureau to approximately reflect real-world conditions. The target variable of this set is medical expenses of each individual. A description for each feature is shown in Table 4.4.

Table 4.4: Medical expenses feature vector

Feature	**Description**
AGE	Patient age
SEX	Patient gender
BMI	Body mass index of patient
CHILDREN	Number of children of the patient
SMOKER	Smoker status (0 if does not smoke, 1 otherwise)
REGION	Region where patient lives (southeast, southwest, northeast or northwest)
CHARGES	Medical expenses incurred (target variable)

The performance of each method using all features is shown in Table 4.5.

WEVREG obtains average results in this set, positioning it at the center of the table. Then for the FS selection methods, the performance of WEVREG with different input vectors was tested. Based on the ranking of features obtained in each FS method, 8 input vectors are used for each method (ranging from 1 feature to the total number of features). In Table 4.8 the performance yield by each input vector selected from the ranking of features by each FS method is shown.

Figure 4.8 shows that the performance is the same for all FS methods. A heatmap of the position given by each method is used to analyze the ranking of the features; it is shown in

Table 4.5: Medical expenses prediction performance

Method	**RMSE**	**MAE**	**MAPE**	R^2
RF	$4,553.73 \pm 890.65$	$2,533.03 \pm 355.15$	0.30 ± 0.01	0.84 ± 0.06
GB	$4,891.33 \pm 739.64$	$2,732.95 \pm 201.07$	0.32 ± 0.04	0.82 ± 0.05
WkNN	$4,959.80 \pm 708.79$	$2,837.28 \pm 325.21$	0.34 ± 0.04	0.81 ± 0.05
ANN	$5,031.20 \pm 742.29$	$2,961.34 \pm 269.63$	0.28 ± 0.02	0.81 ± 0.05
WEVREG	$5,127.74 \pm 553.27$	$3,323.56 \pm 173.47$	0.45 ± 0.03	0.80 ± 0.04
KNN	$5,948.98 \pm 440.82$	$3,714.24 \pm 173.72$	0.38 ± 0.01	0.74 ± 0.03
SVR	$6,007.96 \pm 460.10$	$4,050.73 \pm 219.23$	0.43 ± 0.03	0.73 ± 0.04
RT	$6,899.81 \pm 173.73$	$3,259.43 \pm 184.23$	0.35 ± 0.06	0.65 ± 0.01

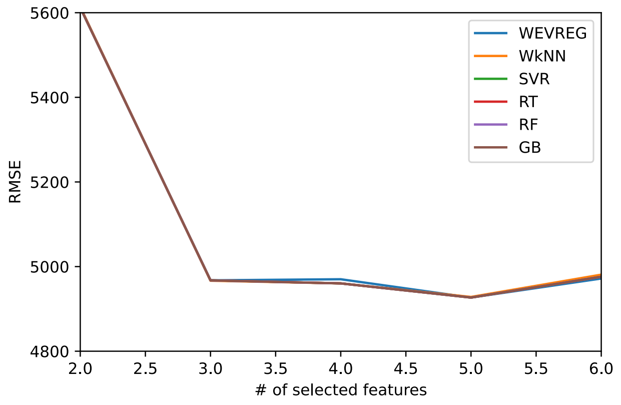
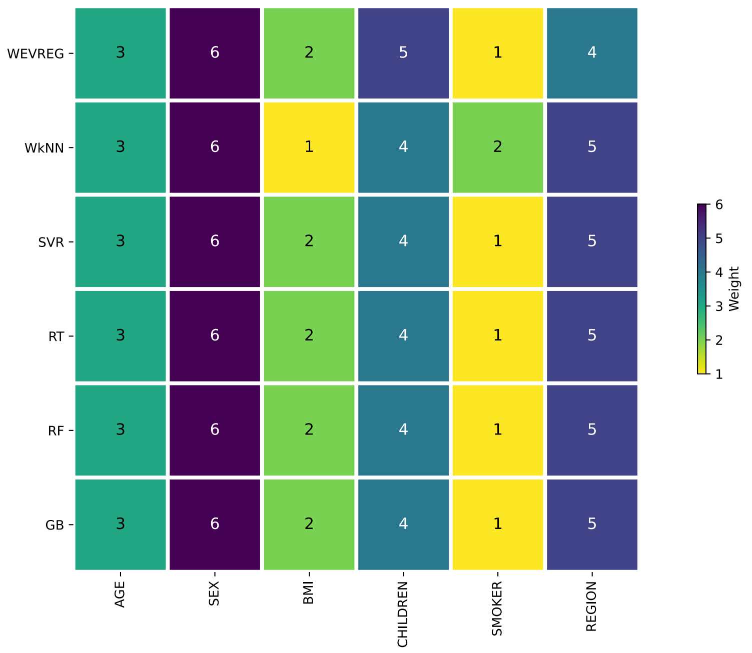


Figure 4.8: Medical expenses feature selection performance

Figure 4.9.



	AGE	SEX	BMI	CHILDREN	SMOKER	REGION
WEVREG	3	6	2	5	1	4
WkNN	3	6	1	4	2	5
SVR	3	6	2	4	1	5
RT	3	6	2	4	1	5
RF	3	6	2	4	1	5
GB	3	6	2	4	1	5

Figure 4.9: Medical expenses feature importance

Figure 4.9 shows that all methods ranked sex and smoker status of each individual as their top two preferences. Results show every method has the same performance, as shown in Figure 4.8. Every method seems to rank the features in the same order, except for WkNN which swaps the most importance features between smoker status and body mass index.

4.2.3 Red wine quality

Red wine quality dataset [68], is data collected for *vinho verde* a Portuguese wine between 2004 and 2007. The set consists of analytical and sensory analysis of red *vinho verde* wine. The goal in this set is to predict the quality of the wine based on its analytical data. Unlike previous datasets, this one can be tackled as a classification problem. The target variable (wine quality) has only 10 possible values, which is equivalent to a classification problem with 10 classes. The predicted outputs in each one of the tested methods for this problem will be a floating point number that will not be rounded. The features vector and description are shown in Table 4.6.

Table 4.6: Red wine quality feature vector

Feature	**Description**
FIXED_ACIDITY	Amount of fixed acids
VOLATILE_ACIDITY	Amount of acetic acid
CITRIC_ACID	Amount of citric acid
RESIDUAL_SUGAR	Amount of remaining sugar after fermentation stops
CHLORIDES	Amount of chlorides in the wine
FREE_SULFUR_DIOXIDE	Amount of free form sulfur dioxide
TOTAL_SULFUR_DIOXIDE	Amount of free and bound forms of sulfur dioxide
DENSITY	Density of the wine
PH	Acidity of the wine in pH scale
SULPHATES	Amount of sulphates
ALCOHOL	% of alcohol content
QUALITY	Wine quality based on physicochemical tests (target variable)

All the available features were used to predict the quality of each wine. The results are shown in Table 4.7

Table 4.7: Red wine quality prediction performance

Method	**RMSE**	**MAE**	**MAPE**	$\boldsymbol{R^2}$
KNN	0.58 ± 0.03	0.40 ± 0.01	0.08 ± 0.00	0.47 ± 0.01
GB	0.59 ± 0.01	0.45 ± 0.00	0.08 ± 0.00	0.45 ± 0.02
ANN	0.62 ± 0.01	0.48 ± 0.01	0.09 ± 0.00	0.41 ± 0.04
RF	0.63 ± 0.01	0.50 ± 0.00	0.09 ± 0.00	0.39 ± 0.02
SVR	0.63 ± 0.02	0.49 ± 0.00	0.09 ± 0.00	0.39 ± 0.01
WEVREG	0.63 ± 0.02	0.51 ± 0.00	0.09 ± 0.00	0.38 ± 0.01
WkNN	0.64 ± 0.01	0.50 ± 0.00	0.09 ± 0.00	0.37 ± 0.03
RT	0.81 ± 0.03	0.50 ± 0.02	0.09 ± 0.00	-0.02 ± 0.12

WEVREG obtained a lower than average performance, as shown in Table 4.7. However, the performance of each method is extremely close to the others. Next, the FS performance was tested using WEVREG as a prediction method. With different input vectors that use an increasing number of features, ranging from 1 to the total number of features. The features of the input vectors are selected based on the ranking of the features obtained from each one of the FS methods. The results are shown in Figure 4.10.

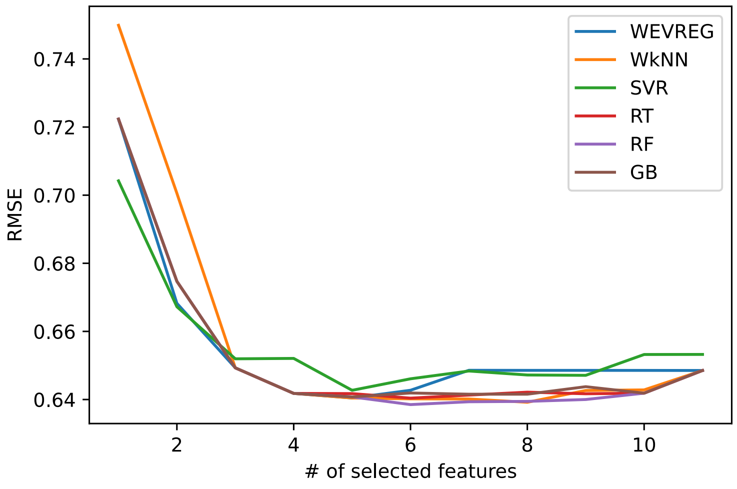


Figure 4.10: Red wine quality feature selection performance

All methods obtain similar performance when selecting more than 3 features, as it can be seen in Figure 4.10. The two configurations of WEVREG achieve slightly worse performances than the tree based algorithms and WkNN for this problem. The top three features selected by every method seem to be the same as all methods share the same performance for the use of three features. The rank obtained by each method is shown in Figure 4.11.

Every method (except for SVR) shares the same top three features, as expected. There is a clear preference for alcohol percentage as the most important feature, which indicates that alcohol percentage is the best indicator for the quality of red wine in this particular set. WEVREG starts to differ from all other methods after the sixth most important feature. That is the explanation for its slightly worst performance, as it can be observed in Figure 4.10 .

4.2.4 Abalone

The Abalone dataset [69] is data collected from physical measurements made to abalones. The age of an abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings using a microscope. Biologists also use measurements such as weight and height to predict the age of an abalone since these measurements are easier to obtain.

The prediction performance using all features is shown in Table 4.9.

WEVREG achieves a poor performance obtaining lower than average performance in this set, even lower than the linear SVR. Then, WEVREG with a variable number of features

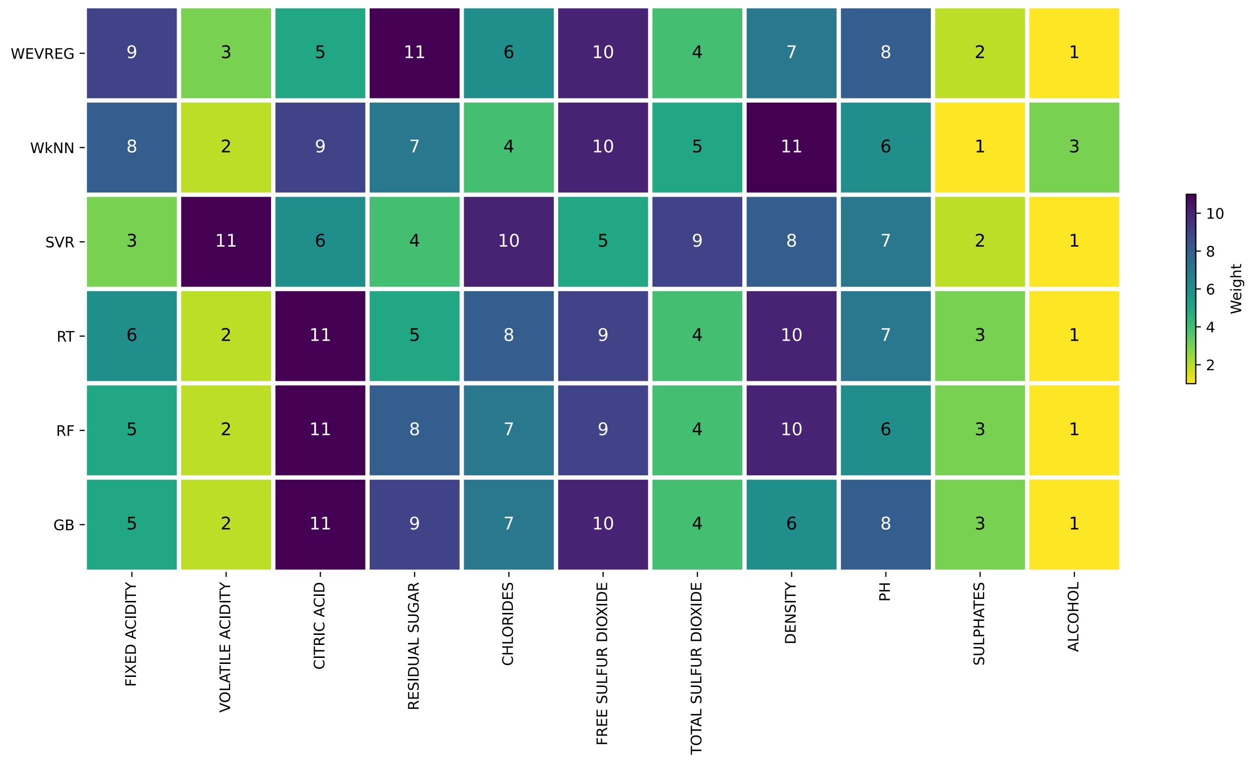


Figure 4.11: Red wine quality feature importance

Table 4.8: Abalone feature vector

Feature	Description
SEX	Male, female or infant
LENGTH	Longest shell measurement
DIAMETER	Diameter perpendicular to length
HEIGHT	Height with meat in shell
WHOLE	Weight of whole abalone
SHUCKED	Weight of meat
VISCERA	Gut weight (after bleeding)
SHELL	Weight after being dried
RINGS	+1.5 gives the age in years (target variable)

was used to test the FS performance by each one of the FS methods. The results are shown in Figure 4.12.

Unlike previous sets, the differences in performances between methods is noticeable in Figure 4.12. WEVREG obtains bad performances when the number of features is under 5. RF is the algorithm able to obtain the best performance by only using 3 of the total of 8 features available for this problem. The rank of the features by each method is shown in Figure 4.13.

As shown in Figure , the methods agree that shell weight is a good indicator of the abalone age. There is a strong disagreement between the methods for other features, thus the results

Table 4.9: Abalone prediction performance

Method	**RMSE**	**MAE**	**MAPE**	$\boldsymbol{R^2}$
ANN	2.16 ± 0.03	1.56 ± 0.01	0.13 ± 0.00	0.56 ± 0.01
GB	2.18 ± 0.02	1.53 ± 0.00	0.13 ± 0.00	0.56 ± 0.00
SVR	2.22 ± 0.03	1.61 ± 0.01	0.14 ± 0.00	0.54 ± 0.01
KNN	2.27 ± 0.01	1.56 ± 0.01	0.13 ± 0.00	0.52 ± 0.00
RF	2.29 ± 0.00	1.62 ± 0.03	0.14 ± 0.00	0.51 ± 0.01
WkNN	2.31 ± 0.07	1.61 ± 0.03	0.13 ± 0.00	0.50 ± 0.02
WEVREG	2.44 ± 0.09	1.75 ± 0.09	0.15 ± 0.01	0.45 ± 0.05
RT	2.94 ± 0.11	2.04 ± 0.05	0.17 ± 0.00	0.19 ± 0.07

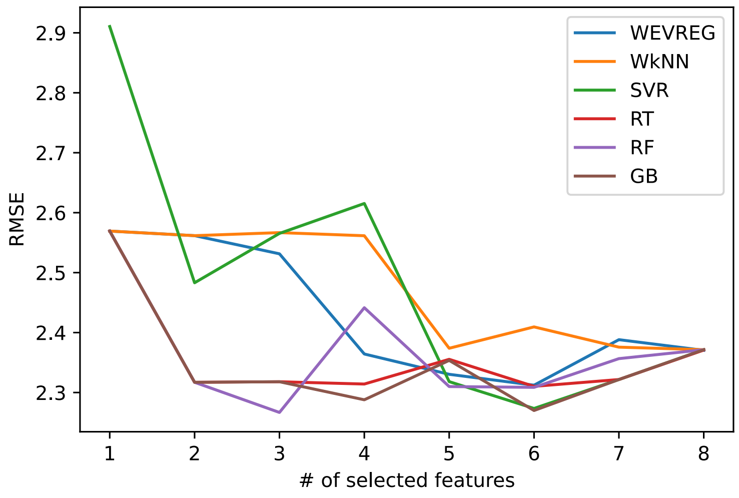


Figure 4.12: Abalone feature selection performance

obtained in Figure 4.12.

4.3 Results

The performance of the proposed method in prediction and FS tasks has been shown. WEVREG can predict and also rank the importance of its features. The proposed method is capable of obtaining comparable results to other well-known regression and embedded methods in prediction and feature selection tasks. The proposed method gets similar results to other well known embedded methods in FS tasks. It is able to detect complex relations better than simple dependencies on synthetic data. It can get similar performance to the best well known embedded methods in well-known datasets. Concerning prediction performance, WEVREG gets performances comparable to the tested methods. These results show the viability of the proposed method as an embedded regression method.

An important concern related to prediction methods is the reliability perception the users may have. Many methods are not trusted by their users. Every machine learning method requires an overall measure of trust to be used afterwards in real-world scenarios. One approach is to use the prediction performance in a validation set. This is a set which the methods have

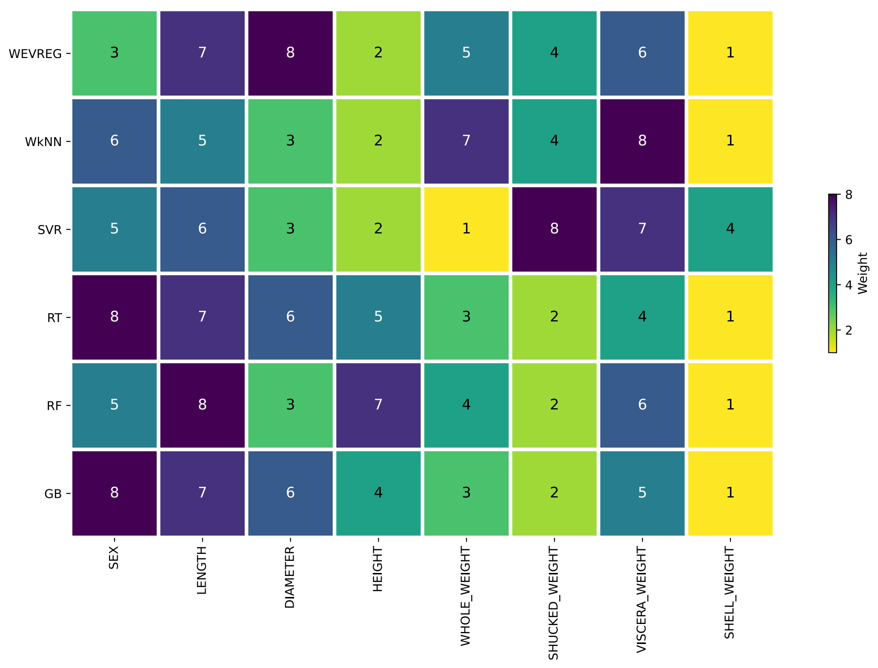


Figure 4.13: Abalone feature importance

not previously seen and has not been used in the training phase of the algorithms. As shown in this chapter, the proposed model can reach decent performances in regression problems, but even if a model achieves a great performance in the validation set, there is no guarantee that it will have the same performance when it is deployed. It is even more problematic when the method is treated as a black box and the model behaviour is not completely understood. When the proposed method makes a prediction, the output is computed based on the mass (importance) of every neighbor near the given input vector. The computation of each mass can be followed (while the number of neighbors is limited), providing a better understanding of the output of the method.

Ribeiro et al. [6] suggested that models could provide explanations for individual predictions in order to increase users' trustfulness. They outlined the desired characteristics of models explanations. The most essential criterion is that they have to be interpretable, i.e., provide a qualitative understanding between the input variable and the response. Another desired characteristic is local fidelity, i.e., the model has a good behavior in the vicinity of the instance predicted. There are methods that are inherently interpretable, such as Linear Regression and Regression Trees. The proposed methods were based on EVREG, due to its transparency and clarity. It also uses a k-NN approach so an explanation for a prediction can be easily obtained. User limitations need to be considered so that the method can be viewed as an interpretable method. For example if the data has thousands of features, then it is difficult for a user to understand their importance for the prediction. So for the proposed method to be considered interpretable, the number of features and neighbors need to be limited to the user capabilities.

Since the proposed method is an embedded method, it is possible to show the global importance of features using each piece of evidence (neighbors and target variable domain) explaining the prediction and uncertainty of the target variable. As an example, let us consider the case of predicting single instance of the Boston housing dataset (subsection 4.2.1). Only the top 5 features are used for reader's simplicity, the input vector with its prediction is shown in Figure 4.10. The true value of this instance is 13.8.

Table 4.10: Input vector and prediction

LSTAT	RM	ZN	PTRATIO	CRIM	Predicted target
37.97	4.14	0.00	20.2	18.50	12.32

From the proposed method, the importance of each feature and the neighbors used can be easily obtained. In Figure 4.14, the boundaries of the neighbors with each dimension importance are shown.

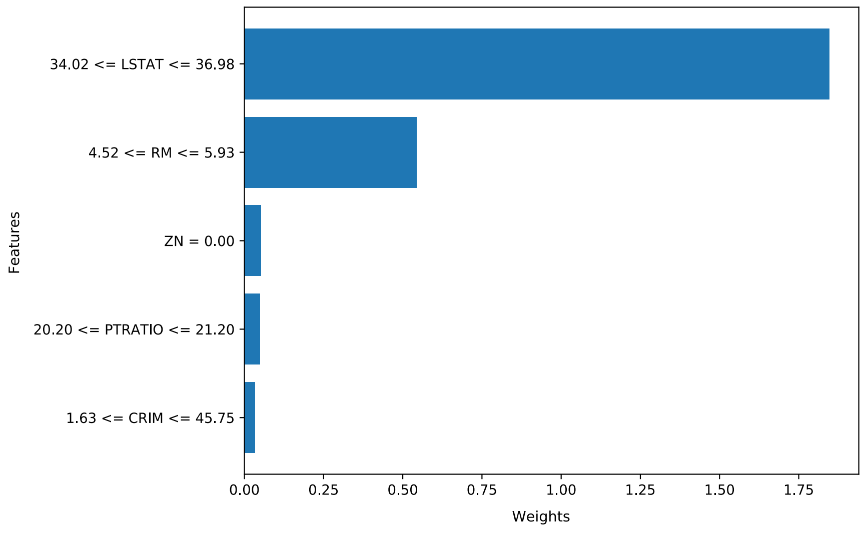


Figure 4.14: Features importance with neighbors boundaries

In Figure 4.14, the vertical axis shows each feature with the boundaries according to the vector neighbors, the horizontal axis represents the importance (weight) of each feature. If further information is needed then the neighbors that were used for the prediction can be presented, showing the effect of each neighbor by its corresponding mass (importance). This can be observed in Table 4.11.

Table 4.11: Neighbors used for the prediction

Mass	LSTAT	RM	ZN	PTRATIO	CRIM	Target
0.14	36.98	4.52	0.00	20.20	45.75	7.00
0.12	34.77	4.91	0.00	20.20	11.11	13.80
0.12	34.37	4.63	0.00	20.20	18.81	17.90
0.12	34.41	5.02	0.00	21.20	1.63	14.40
0.10	34.02	5.94	0.00	20.20	13.68	8.40

The uncertainty of the prediction can also be shown as a function of the target variable domain mass. This information is shown in Table 4.12.

Table 4.12: Model uncertainty for the response

inf $\boldsymbol{y}$	**sup $\boldsymbol{y}$**	**Domain mass**	**Uncertainty**
7.00	17.90	0.40	2.18

As it is shown, the main difference of the proposed method with other well known embedded methods such as GB and RF is its transparency. This difference makes it a preferable method in some use cases. The proposed method will be tested in real world datasets in the next chapters, comparing the method with state of the art algorithms.

Chapter 5

Real case 1: Store Traffic

The proposed method was tested using synthetic data and well-known datasets as presented in the previous chapter. In this one, the method will be tested in a real world setting, particularly a time series dataset of the number of visitors in retail stores.

Retail researchers suggest store traffic is one of the most important drivers of store performance under the right conditions [70, 71, 72]. Store traffic refers to the number of visitors in a store during a period of time. Store traffic is often used as a performance metric, but it is not a reliable performance measure by itself. Retailers need to convert their traffic to sales, this term is known as conversion rate. Conversion rate is the proportion of store visitors who purchase at the store. Depending on the business, this conversion rate will vary. For example, for food retailers it is typically assumed that the conversion rate is close to 100% but for others such as clothes retailers the rate could be a much smaller. The conversion rate for these retailers can be improved by store promotions or by efficient management of labor schedules.

Store labor is the second largest expense of retailers and is one of the keys to increase a store conversion rate. The correct use of the available labor could increase sales, increasing store conversion rates. If store traffic could be precisely predicted, labor schedules could be planned accordingly [73, 74]. Consequently, store traffic forecasting is one of the important forecasting problems in retail. A useful forecast model in the store labor schedules needs to be able to predict long periods of time, e.g., a month. Regardless of the practical applications traffic forecasting has, there is not much research done on this problem because of the usual proprietary nature of the relevant data.

We had access to a vast database of hourly store visitors through FollowUp, a customer experience company. The visitors were registered by cameras placed at store entrances. The company uses this technology in more than $3,000$ stores around the world. In this chapter, the performance of the proposed method will be evaluated in the forecast of store visitors using this data. EVREG in which the proposed method is based, has been used in time series forecasting problem such as machine prognosis obtaining good performances compared to other baseline models. Similar results are expected for this problem. Prediction, FS performance and prediction confidence will be evaluated in this case. The proposed method will be compared with methods previously used in store traffic forecasting. These other methods are the Seasonal Autoregressive Integrated Moving Average (SARIMA) that will be the baseline, Support Vector Regressor (SVR), Random Forest (RF), Gaussian Process (GP) and Recurrent Neural Networks (RNN) architecture, in particular a Long Short-Term Memory (LSTM) neural network.

5.1 Related Work

In particular, the forecast of store traffic is a time series forecasting problem. A time series is a set of observations, where each one is recorded at a specific time [24]. Time series forecasting algorithms have been applied in many areas, e.g., retail traffic forecasting, financial market prediction, weather forecasting and machine prognosis. Time series prediction has been applied in multiple areas, and retail is no exception [73, 75, 76, 77]. Retail traffic forecasting is of significant interest for retailers, Lam et al. [73] highlights the importance of store traffic forecasting and proposes a model for the optimization of store labor schedules based on the expected traffic. The model sets store sales potential as a function of store traffic volume, customer type, and customer response to sales force availability. The models need the expected traffic of the store; to do this, a traffic forecast model is used. This model first uses the log-transformation of the traffic variable to induce a constant variance, then a differencing with one week lag is applied to this transformed variable. Finally, ARIMA components are used to adjust for autocorrelations in the residuals. This model was used for a two week ahead forecast and got a percentage-forecast error of about thirty percent on average.

The most widely used time series model is ARIMA, showing its effectiveness in many use cases [73, 78, 79, 80]. However, it lacks the modeling of non-linear relationships. Even though it does not perform well in some problems, the ARIMA model is usually used as a baseline model to show the performance of other methods. Cortez et al. [76] used the data collected in a period of six months from a facial recognition camera to detect the number of daily visitors along with their gender in a sports store. Using ARIMA as one of their baseline methods, they compared six approaches to predict daily store traffic in three different cases, detecting only males visitors, only females and all visitors, forecasting up to a week ahead. Their input vector included the time of the event, a special daily event (weekend or holiday; major sports or entertainment event), and weather conditions such as maximum wind speed, temperature, humidity, and rain. To test the performance of the models in this problem, they proposed two metrics: the Average Normalized Mean Absolute Error (ANMAE) and a novel metric named the Average Estimated Store Benefit (AESB). In the AESB metric, the manager of the store executes a forecast for N days ahead. Then an alternative plan is set with better management (e.g., promotions to bring more traffic). Each daily error is multiplied by an average cost related to a miss opportunity. Overall [76] showed that the SVR was the best forecasting method, regardless of the visitors' gender and only using time as its input. They used a two-stage approach to get the optimal performance; first, they used a sensitivity analysis proposed in [81] to discard the least relevant time lag and then they did a grid search to find the optimal hyper-parameters.

Abrishami et al. [77] used data from 56 stores of different business such as gyms, coffee shops, restaurants, and bars. This data was collected using wireless access points and it was used to forecast the store traffic one week ahead using a RF and SVR model. This work, unlike [75, 76], discarded the use of weather data because no correlation was observed between traffic data and weather; also its inaccuracy when been forecasted only added error to the prediction. So as an input vector, they used the time of the event, holiday status, special event status, and location as a label (e.g. close to schools and tourist city). As performance metrics, this work used RMSE and MAE to measure performance in the same dataset and MAPE is used for comparison between different datasets. Across all different businesses, the best results were obtained by the SVM model. Afterwards, the authors deepened their study in another work where they tested the performance of a LSTM. The LSTM obtained better performance than their previous work. In this new study, the feature vector consisted of a sequence of length 8 which contained the daily foot traffic from the same day in the previous 8 weeks. The forecast window used was of 1 month, but as they used the data of the previous week it was not possible to use their same feature vector in a real world setting.

5.2 Data and problem description

The data registered by FollowUp consists of the hourly visitors (entrances detected) of retail stores. The goal is to predict the expected visitors in a month period. A month can span from 4 to 6 weeks; in this work a month will be considered only a 4-week period, so the forecasting window will be 28 days. Previous works had found that too small time windows (a week) are often [75, 76, 77] applied in real-world settings; however, stores need a longer window of time to carry out a plan based on expected visitors.

The problem of predicting retail stores traffic is a discrete time series forecasting problem. This is a set of observations x_t (visitors) made at fixed time intervals (e.g day), being recorded at a specific time t [24]. So given a time series $X = \{x_0, x_1, \ldots, x_t\}$, the time series $\hat{X} = \{x_t + 1, \ldots, x_n\}$ with $n > t$ and $[t + 1, n]$ been the forecast window needs to be predicted. For this problem the forecast window will be 1 month, i.e., exactly 4 weeks (28 days).

The data is obtained by cameras that are placed at the entrances of retail stores and is later processed and grouped hourly. The data is accessed from an Application Programming Interface (API) supplied by the company. The API returns the visitors by hour. The data obtained is detailed in Table 5.1.

Column	**Description**	**Type**
store_id	Store identifier.	Integer
time	Time of the event.	Datetime
visitors	Visitors in the last hour.	Integer

Table 5.1: Data obtained from Followup API

The company currently registers the traffic of more than 3,000 stores; a smaller subset will be used in this work. The registered time periods varies from store to store; the majority of the stores have less than a year of data. Obtaining data from different stores in the same period of time is not always possible. An example series of store visitors by hour from August 2015 to August 2019 is shown in Figure 5.1.

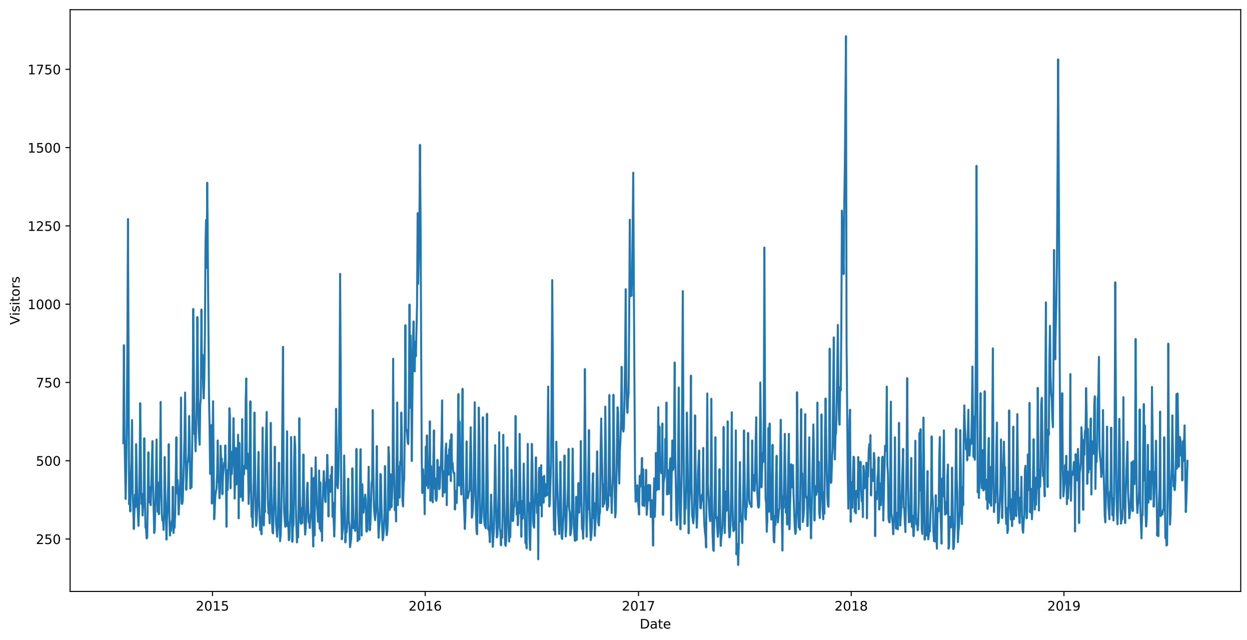


Figure 5.1: Store visitors by date

Since all previous works have used daily data for store traffic forecasting, the data obtained in our project will be also grouped by day. Only 50 stores will be selected, because these ones had the longest history with almost no null values. The window used for the data will be four years, from August 2015 to the same month of 2019. The selection of the feature vector will be detailed in the next section.

5.3 Feature vector

Three feature vectors will be studied and presented for this problem. We will observe their performance and then, just one of them will be chosen to be used in the final experiments.

The first vector will consist of a sequence of previous time steps (visitors) commonly used in time series forecasting. Because the forecast is made with a month window, the values need to have a month of distance. For example if the visitors for August 1st are being predicted, the previous values will consist of July 1st, June 1st and so on. If a shorter window is chosen it would be impossible to create the feature vector for all the dates in the predicted month, because some of their previous values will be unknown. The number of months N for the feature vector will be set based on the proposed method performance. The feature vector with only the sequence of visitors is detailed in Table 5.2.

Column	**Description**
SEQ_1	Visitors 1 Month before
.	.
.	.
SEQ_N	Visitors N months before

Table 5.2: Sequence feature vector

The next feature vector will consist of the same previous sequence with dis-aggregated date features as year, month, day of the week, etc. It is assumed that in some stores the day of the week is highly correlated with the expected number of visitors, e.g., a store which is located near offices is expected to receive high number of visitors during work days. The vector is detailed in 5.3.

Column	**Description**
YEAR	Year number
QTR	Quarter number
MON	Month number
WEEK	Week number
WOM	Week of the month
DOY	Day of the year
DOM	Day of the month
DOW	Day of the week
SEQ_1	Visitors 1 Month before
.	.
.	.
SEQ_N	Visitors N months before

Table 5.3: Time feature vector

However, the features used in Table 5.3 do not reflect the cyclic quality of time features. For example, the day of the week on the features varies from 0 (Monday) to 6 (Sunday). The furthest day from Sunday is Monday, whereas in reality Sunday and Monday are next to each other. Another feature vector will be tested including these cycles. The new embedding has two features representing the vertical and horizontal components in each feature with a cyclic behaviour. The vector is detailed in 5.4.

An experiment was conducted to select the best feature vector and the best length of the sequence. The feature vectors and the output (visitors) were scaled using a minimum/maximum scale between -1 and 1. As shown in Eq. 5.1.

$$X = \frac{X - \inf_{\forall x_i \in X}}{\sup_{\forall x_i \in X} - \inf_{\forall x_i \in X}} - 1. \tag{5.1}$$

The RMSE (Eq. 4.6) of the proposed method was evaluated in each one of the proposed feature vectors. The used sequence length was tested from 0 to 12 for each vector. Data from August 2019 was used as a validation set; all previous data was used as training set. The proposed method was configured with 10 neighbors, iterating for a maximum of 50 epochs with a learning rate of 0.1. The results are shown in Figure 5.5; every shown point is the mean of the 50 evaluated stores. Each store was evaluated 100 times.

The results in Figure 5.5 show that feature vectors that have time features perform considerably better. Surprisingly, the time feature vector in Table 5.3 without the cycling embedding is the one with the best performance. It seems that the store traffic of this set does not have a cyclic behaviour. For example, for the store a Sunday and a Monday are far

Column	**Description**
YEAR	Year number
SIN_QTR	Quarter vertical component
COS_QTR	Quarter horizontal component
SIN_MON	Month vertical component
COS_MON	Month horizontal component
SIN_WEEK	Week vertical component
COS_WEEK	Week horizontal component
SIN_WOM	Week of the month vertical component
COS_WOM	Week of the month horizontal component
SIN_DOY	Day of the year vertical component
COS_DOY	Day of the year horizontal component
SIN_DOM	Day of the month vertical component
COS_DOM	Day of the month horizontal component
SIN_DOW	Day of the week vertical component
COS_DOW	Day of the week horizontal component
SEQ_1	Visitors 1 Month before
.	.
.	.
SEQ_N	Visitors N months before

Table 5.4: Cyclic time feature vector

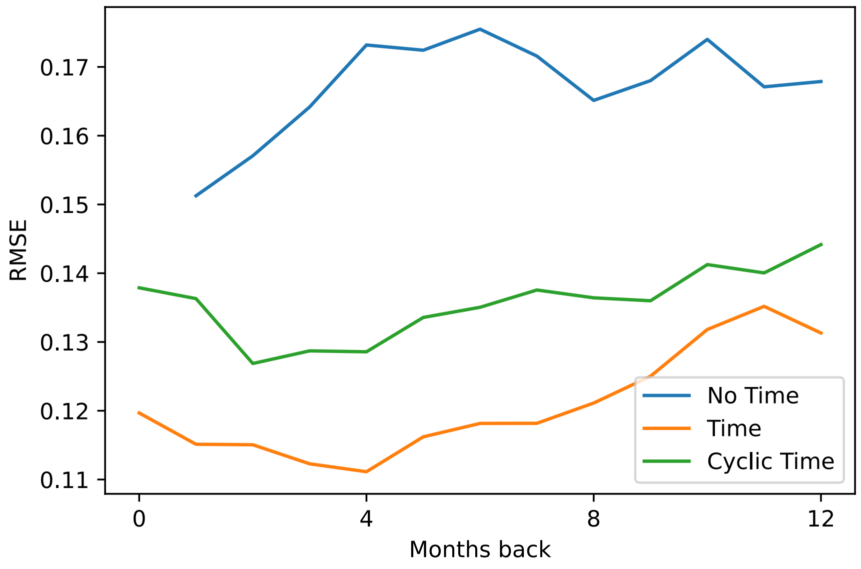


Figure 5.2: Performance of feature vectors (the lower the better)

from each other because the expected traffic differs in a great number. The inclusion of the sequence slightly increases the performance of the method, obtaining the best performance for a sequence of length 4. This means that 4 months of data will be lost in the set, but as it is a small portion of the set, the sequence of length 4 will be used. So the new dataset will consist of daily visitors from December 2015 to August 2019 with an input vector of 12 features.

5.4 Experiments and results

The dataset consists of daily visitors to 50 retail stores from December 2015 to August 2019. Since the forecasting of store traffic is a time series problem, validation methods such as random split or cross-validation cannot be used. The methods performance will be evaluated in a real world setting, with a prediction of one month. Thus the data from December 2015 to July 2019 will be treated as a training set (past observations), and August 2019 will be the validation set (future observations).

For the evaluation of the methods, the performance measures will be calculated by store and then the mean of the 50 stores will be reported. The data for each store will be normalized between -1 and 1 using the same minimum/maximum scale used in Eq. 5.1. In Figure 5.3, the RMSE (Eq. 4.6) for the neighbors configuration of the proposed model is shown.

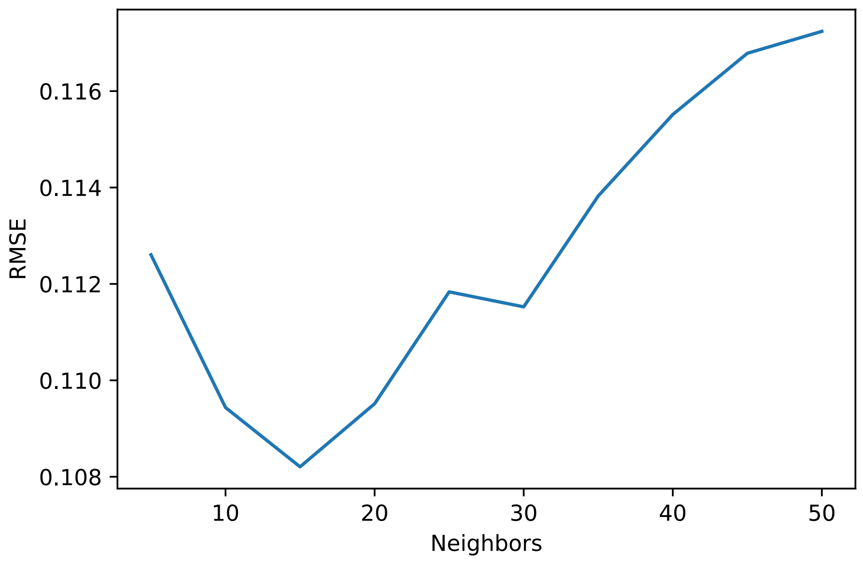


Figure 5.3: RMSE for different neighbors configuration

The number of neighbors in Figure 5.3 has a small effect in the performance of the model. However, the complexity of the algorithm increases with the number of neighbors, and so it is preferable to predict using a small number of them. In this case, for more than 15 neighbors the prediction performance of the methods deteriorates. Therefore the configuration chosen for WEVREG will be 15 neighbors.

The proposed method will be compared with techniques previously used in store traffic forecasting. These are SARIMA, SVR, RF, GP and LSTM neural network. Each method will have a single configuration for the 50 stores. The configuration for most of the methods was reached using a Grid Search approach where the values of the parameters in each method are set based on the ones that yield the best performance (lower mean RMSE) in the forecasting of the 50 stores. The hyper-parameters for the SARIMA method were found using the auto-correlation and partial auto-correlation coefficients. The architecture used for LSTM was as recommended by [82], with a first layer of LSTM cells, a dense layer with an hyperbolic tangent as an activation function to obtain a single output. The configuration of each method is detailed in Appendix A.2.

5.4.1 Prediction performance

The prediction performance will be evaluated using 4 error measures. These measures are RMSE (Eq. 4.6), MAE (Eq. 4.7), MAPE (Eq. 4.8) and a measure previously used in traffic forecasting, the Normalized Mean Absolute Erorr (NMAE). The NMAE between two vectors is calculated as:

$$NMAE(Y, \hat{Y}) = \frac{\frac{1}{n}\sum_{\mathrm{i}=1}^{n} |\hat{y}_{\mathrm{i}} - y_{\mathrm{i}}|}{|\sup Y - \inf Y|}. \tag{5.2}$$

where Y and $\hat{Y}$ are vectors of size n. As shown in Eq. , NMAE is the MAE normalized by the size of the interval of the real response.

The performance of each method was calculated for each store prediction. The prediction of each store was repeated 100 times. Then the mean performance was computed, the results are shown in Table 5.5.

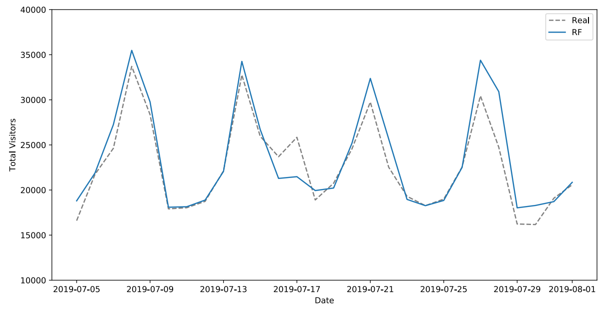
Table 5.5: Stores visitors prediction performance

Method	**RMSE**	**MAE**	**NMAE**	**MAPE**
RF	0.1041 ± 0.01	0.0751 ± 0.02	0.1567 ± 0.06	0.1376 ± 0.04
WEVREG	0.1088 ± 0.01	0.0810 ± 0.02	0.1799 ± 0.10	0.1477 ± 0.04
SVM	0.1133 ± 0.01	0.0843 ± 0.02	0.1756 ± 0.06	0.1538 ± 0.05
GP	0.1321 ± 0.01	0.0972 ± 0.04	0.2010 ± 0.08	0.1817 ± 0.10
LSTM	0.1422 ± 0.02	0.1020 ± 0.03	0.2069 ± 0.06	0.2011 ± 0.09
SARIMA	0.1489 ± 0.01	0.1099 ± 0.03	0.2250 ± 0.06	0.2046 ± 0.08

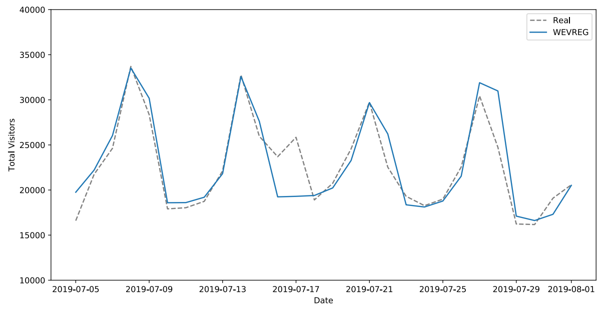
As shown in Table 5.5, RF obtains the best performance as it has been previously reported [77]. The proposed method, WEVREG, achieves good performance obtaining the second place among all compared algorithms. It was expected that LSTM would obtain the best results, but it was only able to outperform the baseline method (SARIMA). The reason behind this poor performance could be the use of a single network architecture for all the stores. Abrishami and Kumar [82] obtain different results compared to the presented one using a specific architecture in each tested store.

The prediction in the experiment was made predicting one store at a time. The data from different stores were not aggregated to make a prediction. Just to observe the overall prediction of each model so that they can be easily compared, the predictions of all stores were aggregated. The aggregated predictions for each method are shown in Figure 5.4.

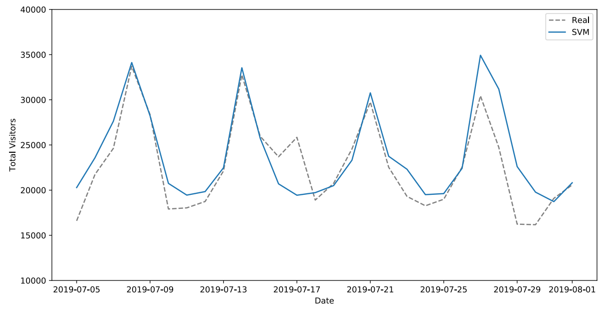
As shown in Figure 5.4, every method is able to detect the base pattern of the data, creating most of the peaks and valleys found on the real data. Methods such as RF and WEVREG are more fitted to the real curve than the others. For the 7th of July, there is an increase in visitors that is not predicted by any of the methods. Some of them, such as RF, GP and SARIMA are able to detect an increase of visitors but the result is of a much smaller size than the real data shows.



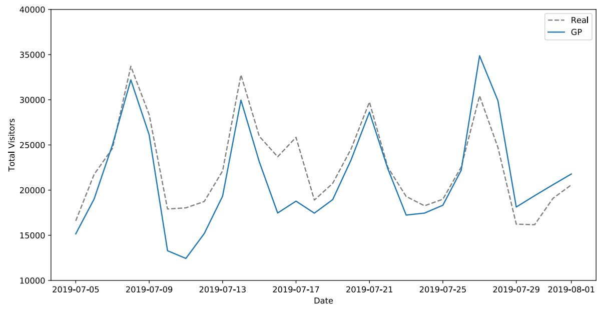
(a) RF



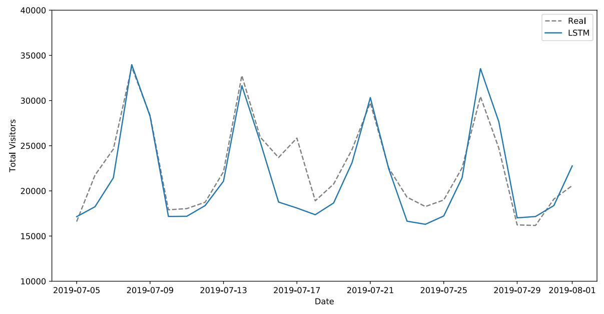
(b) WEVREG



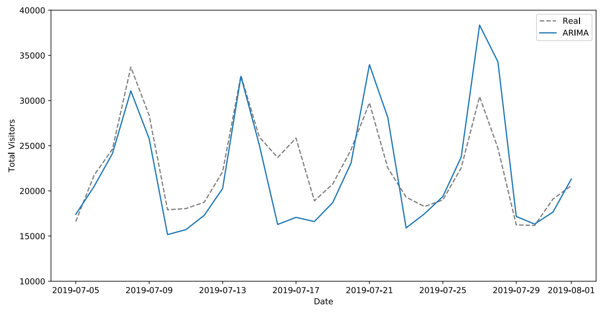
(c) SVM



(d) GP



(e) LSTM



(f) SARIMA

Figure 5.4: Predictions for aggregated visitors

5.4.2 Feature selection performance

Being this a real world setting, the features importance of the proposed vector is unknown. It is assumed that the time features are more important than others. For example, for some stores the day of the week could be more important than the month, because it is expected that stores have more visitors during weekends. Two other embbedding methods were used to test this assumption. These are RF and GB which have been previously compared to WEVREG in FS tasks. The performance of WEVREG is shown In Figure 5.5, selecting a variable number of features based in the ranking created by each method.

The best performance for all methods is reached when all features are used, as shown in Figure 5.5. RF seems to rank features better, always obtaining the lowest error for the features selected, while WEVREG and GB perform similarly.

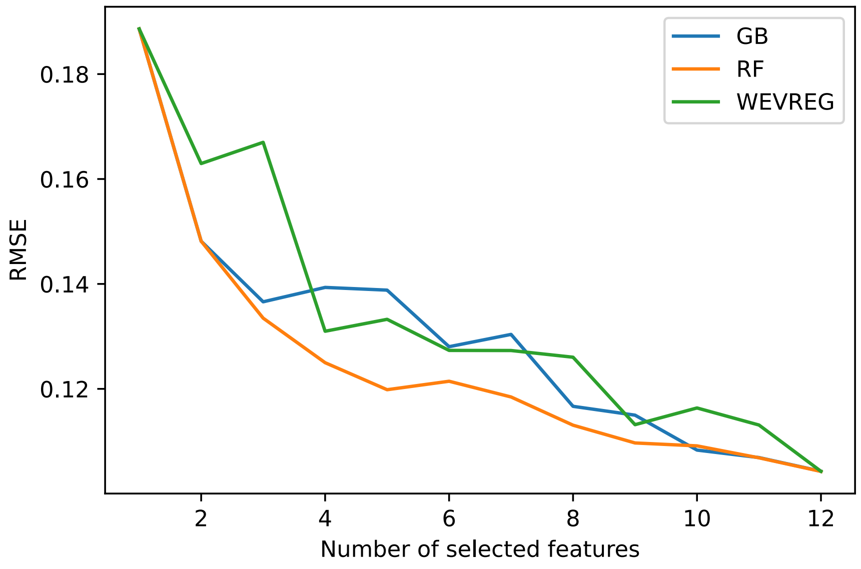


Figure 5.5: RMSE for features selected

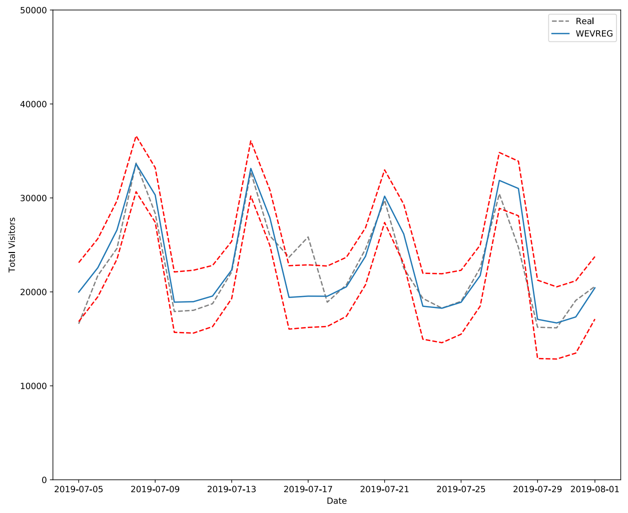
5.4.3 Confidence interval

A unique feature of the proposed method compared to all other embedded methods in this work, is the possibility to generate a confidence interval for each generated prediction. A small example to illustrate the confidence interval of the method will be presented. Two of the methods used for the prediction of store traffic (GP and ARIMA) are able to create a confidence interval of the response. Again the aggregated visitors will be presented to illustrate the confidence interval of these three methods. The aggregated predicted curves with their confidence intervals can be observed in Figure 5.6.

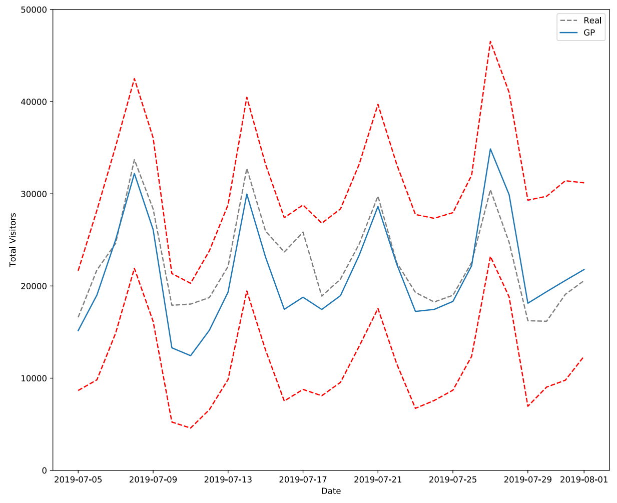
As shown in Figure 5.6, WEVREG and GP have a sharper confidence interval compared to SARIMA, as the intervals are close to each other, because SARIMA is extrapolating the values, so the confidence of the forecast tends to decrease rapidly with each passing day. The proposed method seems to have a more compact interval than GP, but GP has all the real values inside its confidence interval. Ideally the confidence interval needs to contain all the real values while having an interval as sharp as possible. The Relative Interval Size (RIS) [83] metric will be used to measure how sharp (informative) the produced intervals are. The RIS is defined as

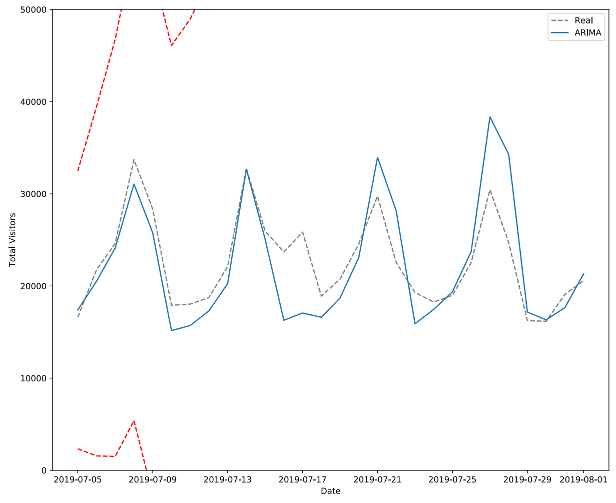
$$RIS(Y) = \frac{1}{N}\sum_{\mathrm{i}=1}^{N}\frac{u_{\mathrm{i}} - l_{\mathrm{i}}}{\sup Y - \inf Y} \tag{5.3}$$

where u_{i} and l_{i} are the upper and lower bounds of the confidence interval of output i. $\sup Y$ and $\inf Y$ are the maximun and minimum of the response interval. To quantitatively measure the characteristics desired in a confidence interval a new measure will be introduced. The new measure needs to reward the accuracy and penalize the sharpness (Eq.5.3) of the response interval. Based on this, a new measure named Confidence Interval Performance (CIP) is defined as:



(a) WEVREG





(b) GP

(c) SARIMA

Figure 5.6: Aggregated prediction with confidence interval

$$CIP(Y,\hat{Y}) = \frac{N_{\text{i}}}{N_t} - \frac{sharp(\hat{Y})}{|\sup Y - \inf Y|} \tag{5.4}$$

where Y is the vector of the real values, $\hat{Y}$ is the predicted vector. N_{i} is the number of real values inside $\hat{Y}$ confidence interval and N_t is the total points of vector Y. CIP is equal to 1 when the predicted curve matches exactly the real one, and the interval is of size 0. So CIP is bounded to interval $]-\infty, 1]$ where the higher the better.

The CIP of WEVREG, GP and ARIMA was evaluated. The forecasting of each method was computed 25 times and the mean CIP was calculated. The results are shown in Table 5.6.

Table 5.6 shows that WEVREG and GP obtain similar performance with the first one

Method	**CIP**
WEVREG	0.16 ± 0.03
GP	0.12 ± 0.00
ARIMA	-2.87 ± 0.00

Table 5.6: Confidence Index

having a slight advantage. Even though WEVREG misses some of the real values as it can be seen in Figure 5.6, the interval has a considerable smaller size, which gives it the edge to obtain a better result.

Chapter 6

Real Case 2: Health Care Costs

In this chapter, the proposed method will be tested on a health care costs prediction problem. This problem concerns the prediction of future health care costs of an individual based on their past medical and billing information. The data was provided by Tsuyama Chuo hospital, a hospital located in Tsuyama, Okayama Prefecture, Japan.

Health care expenditure is one of the most critical issues in today's society. World Health Organization (WHO) statistics show that global health care expenditure was approximately US$ 7.5 trillion, equivalent to 10% of the global GDP in 2016 [84]. One of the reasons for these high expenses in care is the low accountability in health care in some countries, for example, inefficiencies in the US health care system result in unnecessary waste, provoking a large discrepancy between spending and returns in care [85].

If health care costs for each patient could be predicted with high certainty, problems such as accountability could be solved, enabling control over all parties involved in patients' care. It could also be used for other applications such as risk assessment in the health insurance business, allowing competitive insurance premiums, or as input information for developing new government policies to improve public health.

With the current frequently use of electronic health records (EHRs), an interest has emerged in solving accountability problems using data mining techniques [86]. There have been various approaches to predict health care costs for large groups of people [87, 88]. On the contrary, prediction for an individual patient has rarely been tackled. Initially, rule-based methods [89] were used for trying to solve these problems requiring domain knowledge as if-then rules. The downside of this method is the requirement of a domain expert to create the rules, thus making the solution expensive and limited to the dataset being used. In the current state-of-the-art, statistical and supervised learning methods are preferred with the latter getting the best performance. The reason for best performance is the skewed and heavy right-hand tail with a spike at zero present in the distribution of health care costs [90].

Previous works in health care costs prediction [91, 92, 93] have reached to the conclusion that clinical features yield the same performance as using only cost predictors without a proper FS experiment. One aim of this chapter is to prove these claims with a proper FS method. Japanese health records from Tsuyama Chuo Hospital were used to test this claim. These records include medical checkups, exam results, and billing information from 2013 to 2018. They were used to compare the proposed method performance with the other less transparent methods such as Artificial Neural Networks (ANNs) and Gradient boosting (GB).

The proposed method will be used to select features and predict the health care costs of individuals in this chapter. The performance of WEVREG will be compared with four supervised learning algorithms previously reported for this problem.

6.1 Related Work

Health care costs for a group of people commonly have a spike close to zero and a skewed distribution with a heavy right-hand tail; statistical methods trying to predict health care costs suffer from this characteristic in small to medium sample sizes [94]. Advanced methods have been proposed to address this problem, for example, Generalized Linear Models (GLMs) where a mean function (between the linear predictor and the mean) and a variance function (between the mean and variance on the original scale) are specified and the parameters are estimated given these structural assumptions [95]. Another example is the two-part and hurdle model, where a Logit or Probit model is used first to estimate the probability of the cost being zero, and then, if it is not, a statistical model such as log-linear [96] or GLM is applied. The most complex statistical method used to solve this problem is the Markov chain model; Marshall et al. [97] suggested to estimate the resources use over different phases of health care. Mihaylova et al. [98] present a detailed comparison of statistical methods in health care cost prediction.

On the other hand, supervised learning algorithms have been extensively used to predict health care costs; the type of data used for these methods vary. While a few works use only demographic and clinical information (e.g., diagnosis groups, number of admissions and number of laboratory tests) [99], the majority have incorporated cost inputs (e.g., previous total costs, previous medication costs) as well [91, 100, 92, 93], obtaining better performance. GB [101] excels as the method with the best performance for this problem [93], which is an ensemble-learning algorithm, where the final model is an ensemble of weak regression tree models, which are built in a forward stage-wise fashion. The most essential attribute of the algorithm is that it combines the models by allowing optimization of an arbitrary loss function, i.e., each regression tree is fitted on the negative gradient of the given loss function, which is set to the least absolute deviation [102]. ANNs come close to the performance of GB. An ANN is an extensive collection of processing units (i.e., neurons), where each unit is connected with many others; ANNs typically consist of multiple layers, and some goal is to solve problems in the same way that the human brain would do it [103]. Another type of model with good results is the M5 Tree [92]; this algorithm is also a Regression Tree (RT), where a Linear Regression Model is used for building the model and calculating the sum of errors as opposed to the mean [104].

It has been discovered that most health care expenses of a population are generated by a small group of people, as Bertsimas et al. [91] showed in their dataset: 80% of the overall cost of the population originates from only 20% of the most expensive members. Therefore, a classification phase is suggested to classify patients in a risk bucket when trying to improve the performance of the methods listed above. Morid et al. [90] reported that for low-risk buckets, GB obtains the best results, but for higher ones, ANN is recommended. It has also been found that costs rise sharply with nearness to death [105, 106, 107]. These approaches will not be used as they lie outside the scope of this work.

6.2 Data and Problem Description

The problem addressed in this chapter is predicting future health care cost of individuals, using their past medical and cost information. This is a supervised learning problem, which can be formally specified as a regression problem where the input vector $x = (x_0, x_1, \ldots, x_n)$ is an individual's past medical and cost information and the target variable y is that person's health care expenses in a future period (e.g., a year). The records used for this work were provided by Tsuyama Chuo Hospital, a Japanese hospital located in Okayama Prefecture. These records were gathered between 2013 and 2018.

Japan has universal coverage for social health insurance; the system is composed of three sub-systems, National Health Insurance (self-employment), Society Health insurance (for employees) and a Special Scheme for the elderly (75 or older). Every citizen must join one of these three sub-systems according to their occupational status and age. The premium charge for each person is set by each insurer depending on the person's income. Each medical organization is paid by a fee-for-service principle; at the end of the month every medical facility in Japan has to send a set of claims to be reimbursed by an insurer (as a claim sheet); the insurers have the right to decline a claim if it is incorrect or seems unnecessary.

Medical facilities use a special software for the production of a claim sheet. This software registers all procedures, drugs and devices for each patient. Each procedure has a standard code set by the Ministry of Health, Labor and Welfare (MHLW) that can be translated directly to the International Statistical Classification of Diseases and Related Health Problems codification (ICD-10) [108], a medical classification list created by the WHO.

Every claim sheet sent by health facilities all over Japan is gathered by the MHLW in a National Database [109]. The database contains a detailed information on patients such as provided service, age, sex, date of consultation, date of admission, date of discharge, procedures and drugs provided with volumes and tariffs. 1700 million records are registered annually. The data to be used contains the electronic claims (claim sheets) sent by Tsuyama Chuo hospital to the National Database between 2013 and 2018.

The claims are stored in a set of files; these files had to be transformed to study the data. The format of the claims file is confusing without previous knowledge of the structure. The detailed documentation of this format can be found at the Medical Remuneration Service website (`http://www.iryohoken.go.jp/shinryohoshu/file/spec/22bt1_1_kiroku.pdf`). A short summary of the file format is shown in Table 6.1.

Since a patient's data is dispersed within these files, this raw data could not be used to predict the health care expenditure of patients. The data in these files was used to create a patient's representation that could be used for the prediction of an individual's health care costs. Each patient in the insurance claims could be identified by a unique identifier, which

Table 6.1: Insurance claims

Header	Name	Description
IR	Medical institution	Details of the medical institution.
RE	Insured details	Patient details with dates and demographics.
HO	Insurer details	Patient insurer information.
KO	Public expenses	Patient public expense information.
KH	Special information	Patient especial information (free text).
SY	Diagnosis	Patient diagnosis in MHLW coding.
SI	Procedure	Details for a patient treatment.
IY	Medications	Details for the medicines given.
TO	Specific equipment	Specific equipment details used in a patient.
CO	Comment	Comments for diagnoses or symptoms (free text).
SJ	Symptoms	Patients symptoms.

makes it possible to follow all patients throughout the years.

Besides the data sent by Tsuyama Chuo hospital to the National Database, patients' health checkups information was available. Every Japanese worker needs to take a yearly health checkup to start or continue working at a company, so there were many patients with this data available.

The available data had patients' monthly history between 2013 and 2018. However, there were many missing values because most patients had few claims each year. Therefore, claims were grouped yearly so that we could have fewer missing values. Three different scenarios were created for this purpose as follows. **Scenario 1**: A single year of history to predict the next cost value. **Scenario 2**: Two years of data to predict the third one, and **Scenario 3**: Five years of history to predict the costs of the sixth year. If they did not have the required history available patients will be filtered out, and thus, the sets shrank in each scenario.

Health records are highly susceptible to inconsistent and missing data. The steps to decrease the inconsistencies of the data to a minimum will be briefly described. The dataset has patients demographics, diagnosis, medications, health checkups and billing information from electronic claims that Tsuyama Chuo hospital reports to the Japanese government. For demographics there are no patients with missing gender or birth date. Missing diagnosis can be found in some records for two reasonS, the first being the missing MHLW code, the latter being the inexistence of a ICD-10 translation of the MHLW code. This missing data could not be filled with the median being this a categorical value, so missing diagnosis codes will be skipped.

Concerning medication records, missing values can be found on the times the medication was administered and medication score. The missing data could be filled with zeros or the median for the times the medication was administered, but on the case of the medication score this variable has a direct relation to the total score that is going to be predicted; total score is the sum of all scores including procedures and medications. So if a patient has a missing medication score then the claim which the medication record is related to will not be considered for the set. In the case of missing health check values, the median of the exam was preferred to replace null values. Finally on billing information, any missing value will make the related claim be skipped as well. After decreasing inconsistencies, the basic statistics of the data in each scenario are shown in Table 6.2.

Table 6.2: Statistics of patients' records in each scenario.

Statistics	**Scenario 1**	**Scenario 2**	**Scenario 3**
Total number of patients	71,001	33,646	8,810
Mean costs	11,030	11,536	12,420
Mean age	54.00	58.00	63.00
% Male	48.81	48.54	48.56
% Female	51.19	51.46	51.44

6.3 Feature Vector

The data available is the set of claims sent by Tsuyama Chuo hospital to the National database and patients' yearly health checkups, as mentioned in Section 6.2. Data from both sources were crossed to obtain:

- **Demographics**: Patients' gender and age.
- **Patients' attributes**: General information about patients such as height, weight, body fat and waist measurement.
- **Health checks**: Results from health check exams a patient had undergone. Japanese workers undergo these exams annually by law. Each exam is indexed by a code, and the result is also included. Some examples are creatinine levels and blood pressure. There were 28 different types of exams, and the date when they were collected was also included.
- **Diagnosis**: Diagnosis for a patient illness registered by date and identified by their ICD-10 codes.
- **Medications**: Detailed information of the dosage and medicine administered, including dates and charges.
- **Costs**: Billing information of each patient's treatment. Including medicine, procedures and hospitalization costs. The sum of these costs in a year is the value that was predicted.

The goal of this experiment is to test the performance of the proposed method. Comparing its results to the most successful models reported by the up-to-date literature in terms of prediction accuracy. In this chapter the costs of each patient in the future year need to be predicted. Figure 6.1 shows the distribution of patients' costs. The chart shows that costs had the same distribution as described in [90], with a spike at 0 and a long right-hand tail.

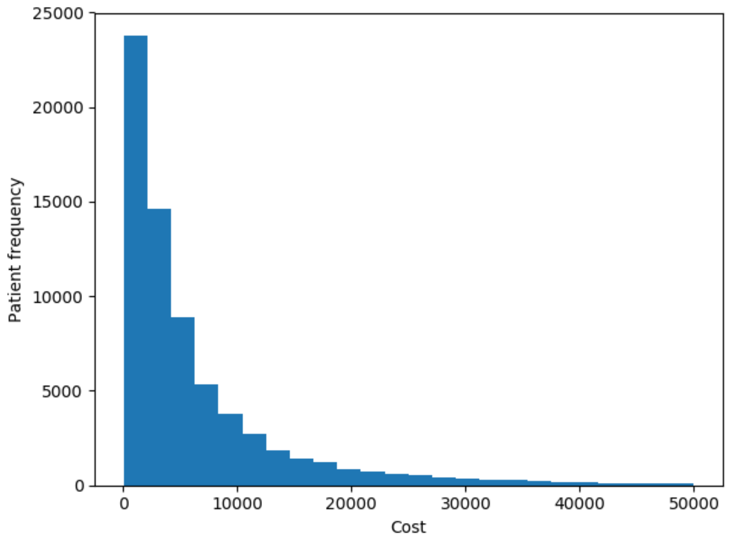


Figure 6.1: Patients costs distribution.

It has been reported [91, 92, 93] that the use of clinical features (health checks, diagnosis and medications) yields the same performance as using only cost features. This conclusion has been reached without the proper use of an FS algorithm, just by testing the use and omission of clinical data, thus it is not clear whether or not specific clinical data helps to determine a patient's costs. Clinical features will be used to properly test this hypothesis.

Encoding a patient's history was done by using all sources available as features. The sources are demographics, health checkups, diagnosis in ICD-10 codification, previous and actual costs. Only chronic conditions will be considered for the diagnosis as these can be carried from one year to the next. The chronic conditions used will be the diseases defined by Koller et al. [110] in a study of the impact of multi-morbidity on long-term care dependency. In the study they used 46 chronic conditions based on ICD-10 codes defined by the Central Research Institute of Statutory Ambulatory Health Care in Germany. Table 6.3 shows a description of the patient's vector representation with the number of variables of each type in each scenario. Each variable that can change from one year to the other, e.g., costs, are represented as a sequence. As an input vector, all dimensions shown in Table 6.3 were used except for the actual cost that was used as target variable.

Table 6.3: Number of variables by type in patient encoding.

Description	**Scenario 1**	**Scenario 2**	**Scenario 3**
Demographics	2	2	2
Health checkups	27	54	135
Chronic diseases	46	92	230
Medication info	2	4	10
Previous costs	1	2	5
Actual cost	1	1	1

6.4 Experiments and Results

The evaluation of the performance of the proposed method was done by comparing its results with the methods reported by Sushmita et al. [92], Morid et al. [90] and Duncan et al. [93] for the health cost prediction problem; these works used RT, GB and ANN methods respectively. Also a similar embedded regressor algorithm will be tested: WkNN.

The MAE (Equation (4.7)) was used to measure the performance of each method. However, the MAE is useful to compare algorithms in the same set but not to compare results in different datasets, so we also used the Mean Absolute Percentage Error (MAPE) (Equation (4.8)). One disadvantage of this error measure is that it does not support zero values in the output of a dataset. It is expected that most individuals do not incur in any cost (healthy individuals). This is not the case for this dataset, as the only information avaliable is of individuals that have concurred to Tsuyama Chuo Hospital for treatment or for a health check for which they have paid. Therefore, MAPE can be used as a performance measure. R^2 was also used which is the Pearson correlation between the predicted and actual health care cost (4.9)).

As recommended by Diehr et al. [88] the target variable (actual costs) was transformed to its logarithmic value in each one of the scenarios, so the distribution of patients' costs in Figure 6.1 was distributed as shown in Figure 6.2.

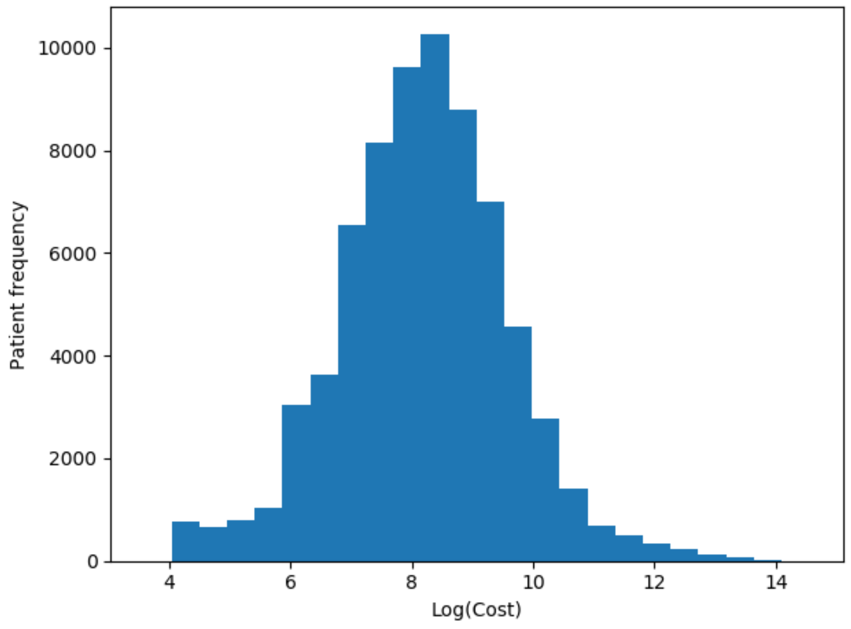


Figure 6.2: Patients logarithmic costs distribution.

The configuration for each method in every scenario is shown in Apendix A.3. A 5-fold cross validation procedure was used to evaluate the performance of each model [111]. The 5-fold cross validation is a statistical procedure where the dataset is randomly divided into five groups, then one of these groups is treated as a validation set and the other four become the training set. This validation is then performed with every group for a total of five times. Finally, the mean performance for each validation group is calculated to obtain the performance of each algorithm. The 5-fold validation was performed five times (resulting in 25 validations) for every tested method. The median of each performance measure for each scenario are shown in Table 6.4, 6.5 and 6.6 with their corresponding standard deviation.

Table 6.4: Models performance scenario 1.

Method	**MAE**	**MAPE**	$\boldsymbol{R^2}$
RT	0.97 ± 0.01	0.13 ± 0.00	0.16 ± 0.01
WkNN	0.95 ± 0.01	$\mathbf{0.12} \pm 0.00$	0.21 ± 0.01
ANN	0.92 ± 0.01	0.13 ± 0.00	0.22 ± 0.01
WEVREG	0.92 ± 0.02	$\mathbf{0.12} \pm 0.00$	0.23 ± 0.02
GB	$\mathbf{0.89} \pm 0.02$	$\mathbf{0.12} \pm 0.00$	$\mathbf{0.26} \pm 0.02$

Table 6.5: Models performance scenario 2.

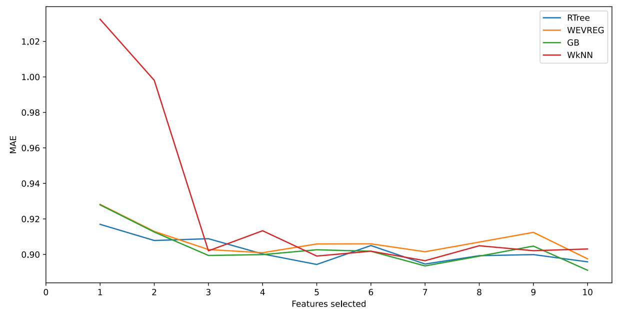
Method	**MAE**	**MAPE**	$\boldsymbol{R^2}$
RT	0.94 ± 0.01	0.12 ± 0.00	0.18 ± 0.02
WkNN	0.91 ± 0.00	0.12 ± 0.00	0.25 ± 0.00
ANN	0.85 ± 0.01	$\mathbf{0.11} \pm 0.00$	0.30 ± 0.01
WEVREG	$\mathbf{0.84} \pm 0.00$	$\mathbf{0.11} \pm 0.00$	$\mathbf{0.33} \pm 0.00$
GB	$\mathbf{0.84} \pm 0.01$	$\mathbf{0.11} \pm 0.00$	0.32 ± 0.01

Table 6.6: Models performance scenario 3.

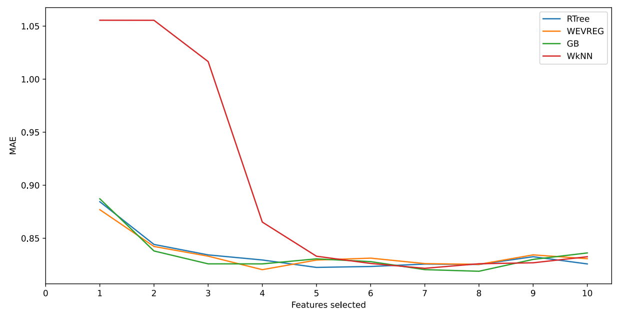
Method	**MAE**	**MAPE**	$\boldsymbol{R^2}$
RT	0.91 ± 0.02	0.11 ± 0.00	0.17 ± 0.02
WkNN	0.82 ± 0.01	0.10 ± 0.00	0.34 ± 0.01
ANN	0.79 ± 0.03	0.10 ± 0.00	0.35 ± 0.04
WEVREG	0.75 ± 0.02	$\mathbf{0.09} \pm 0.00$	0.41 ± 0.02
GB	$\mathbf{0.67} \pm 0.02$	$\mathbf{0.09} \pm 0.00$	$\mathbf{0.49} \pm 0.02$

Every method increased its prediction performance through every scenario even though the sets shrank in every step; this could be attributed to the availability of longer history for every patient in the sets. RT was the only method that did not increase its performance drastically with each scenario, obtaining the overall worst performance. The best results across all scenarios were obtained by GB. WEVREG obtained similar results to GB, even obtaining a slightly better R^2 in the second scenario, but it had a noticeable difference in the last scenario that could be attributed to the smaller size of the set. WkNN obtained worst metrics compared to WEVREG across all scenarios, although it used the same similarity function and similar weight learning. The higher complexity of the masses computation in the proposed method seemed to benefit its prediction power compared to WkNN.

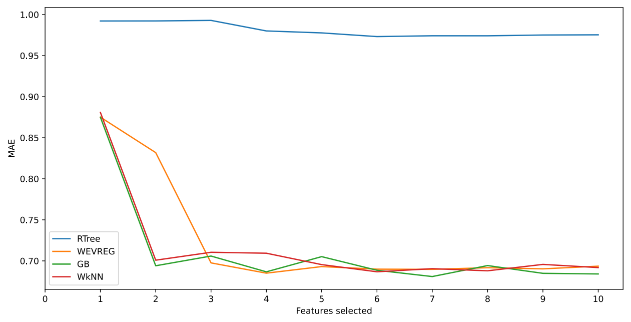
The most important characteristic of the proposed method is its FS capabilities. Weights for each input feature were learned during its training phase. Each feature starts with an initial weight of 1, and it is updated in every iteration of the learning phase. These weights represent the importance of each feature to predict the target variable, which in this case is patients' future health care costs. All the embedded methods tested for this problem were used to compare the FS performance of the proposed method. With the ranking of features computed by each method, the proposed method was used to predict the output with a variable number of these features. The performance (MAE) for the features selected by the methods in each scenario can be seen in Figure 6.3, the lower the curve the better is its FS performance.



(a) Scenario 1



(b) Scenario 2



(c) Scenario 3

Figure 6.3: MAE for different features selected

As it can be observed in Figure 6.3, every method except for RT obtains similar performances when more than 5 features are selected in any scenario. In the first scenario, the optimal number of features to select seems to be 3. WkNN is the only method that does not identifies the most important features in the correct order. So the proposed method needs to select the top 3 features ranked by WkNN to reach the same performance it obtains with the features selected by the other embedded methods. For scenario 2, the best performance seems to be reached with the top 4 selected features. Again WkNN is the method that is slower to reach the performance of every other method. This time, when more than 5 features is selected the same performance is obtained by every method. In the last scenario, 4 features are needed to reach a good performance. In this scenario, the proposed method seems to not detect the correct order of the top features. It is also observed that RT fails to identify any important feature. This could occur because of the high dimensionality of the last scenario. The features selected by the proposed method will be the top five features on each scenario, the features are shown in Table 6.7.

Table 6.7: Top 5 features for each scenario.

Scenario 1		**Scenario 2**		**Scenario 3**	
Feature	**Weight**	**Feature**	**Weight**	**Feature**	**Weight**
cost_ 1	65.06	*cost_ 1*	19.67	*cost_ 5*	19.66
age	6.19	*cost_ 2*	16.47	*cost_ 4*	19.66
gender	1.09	*age*	3.09	*cost_ 3*	19.66
dementia_ 1	1.03	*urinary_ incontinence_ 1*	1.70	*cost_ 2*	19.66
parkinsons_ disease_ 1	1.03	*dementia_ 2*	1.59	*diabetes_ mellitus_ 3*	2.43

As it can be observed in Table 6.7, the most important features across each scenario were cost features. Larger weights are assigned to the closest previous costs.These results are in line with previous works [91, 92, 93], where it was reported that cost features alone are a good indicator for future health expenses.

Patient's age seemed to be a small differentiable feature, that had a small effect in the search of similar patients. Diagnosis features and health checkups features did not have a noticeable effect for health care costs prediction. In the case of the last scenario, even though it was overshadowed by cost features, diabetes mellitus was a meaningful diagnosis to reach a correct cost prediction. This could be related to the fact that patients with this diagnosis incurred high expenses attributed to inpatient care [112, 113]. An important feature across all scenarios was the diagnosis of dementia in the previous year; it seemed to be a small factor to group patient with similar costs in cases where it was present.

The weights learned by the proposed model were used to filter out unnecessary features. The top 5 features in every scenario were used. The performance of the methods were tested once again using these selected features. The results can be seen on Tables 6.8, 6.9 and 6.10.

Table 6.8: Models performance scenario 1 with top 5 features.

Method	**MAE**	**MAPE**	$\boldsymbol{R^2}$
RT	0.92 ± 0.02	0.13 ± 0.00	0.22 ± 0.03
WkNN	0.91 ± 0.01	$\mathbf{0.12} \pm 0.00$	0.23 ± 0.01
ANN	0.92 ± 0.01	0.13 ± 0.00	0.22 ± 0.01
WEVREG	0.91 ± 0.01	$\mathbf{0.12} \pm 0.00$	0.23 ± 0.01
GB	$\mathbf{0.91} \pm 0.01$	$\mathbf{0.12} \pm 0.00$	$\mathbf{0.24} \pm 0.01$

Table 6.9: Models performance scenario 2 with top 5 features.

Method	**MAE**	**MAPE**	$\boldsymbol{R^2}$
RT	0.87 ± 0.03	0.12 ± 0.00	0.27 ± 0.03
WkNN	$\mathbf{0.82} \pm 0.01$	0.11 ± 0.00	0.34 ± 0.01
ANN	0.83 ± 0.02	$\mathbf{0.11} \pm 0.00$	0.32 ± 0.02
WEVREG	$\mathbf{0.82} \pm 0.00$	$\mathbf{0.11} \pm 0.00$	$\mathbf{0.34} \pm 0.00$
GB	$\mathbf{0.82} \pm 0.01$	$\mathbf{0.11} \pm 0.00$	0.33 ± 0.01

Every model, except for GB, increased its performance in all scenarios as shown in Tables 6.8, 6.9 and 6.10. In particular, WkNN and RT had a significant improvement in its MAE and

Table 6.10: Models performance scenario 3 with top 5 features.

Method	**MAE**	**MAPE**	$\boldsymbol{R^2}$
RT	0.81 ± 0.02	0.10 ± 0.00	0.32 ± 0.02
WkNN	0.76 ± 0.02	0.10 ± 0.00	0.41 ± 0.02
ANN	0.72 ± 0.03	0.09 ± 0.00	0.44 ± 0.03
WEVREG	0.72 ± 0.02	0.09 ± 0.00	0.44 ± 0.02
GB	$\mathbf{0.68} \pm 0.02$	$\mathbf{0.08} \pm 0.00$	$\mathbf{0.48} \pm 0.02$

R^2 scores. ANN and WEVREG had a high improvement in the last scenario, which could be attributed to the large number of unnecessary features in the first place. The initial features seemed to induce too much noise for the models to make precise predictions. On the other hand, GB did not seem to be benefited as the other models with the filtered features since it obtained similar performances, showing the GB resilience in the presence of noise in its features compared to the other tested models. These results show the capability of WEVREG to complete FS tasks, allowing us to decrease the number of features considerably, speeding up training times and model performance. They also show that the most important features for health care costs prediction were cost features; other features such as demographics and some diagnosis such us diabetes mellitus helped to determine a patient's costs but were not as indicative as previous costs.

Chapter 7

Conclusions and future work

There has been a growing trend of using black box methods on every supervised learning problem that is encountered. This trend has generated serious problems in some high-stakes use cases. Because the inner workings of these algorithms is unknown, strong biases have been detected in areas such as health care, criminal justice and finance [2] after the methods have already been deployed. Therefore, in recent years, there has been a growing interest in studying and using more transparent methods for these problems, In this work, a new transparent regression method was presented. The proposed method was able to predict and rank features of its input, making it a viable embedded regression method.

The method is based on EVREG, a transparent regression method which uses the Dempster-Shafer Theory. Our aim was to make the method more explainable, i.e., to present its prediction in an understandable manner to a human being. Thus, modifications to EVREG were introduced so that the method was capable of ranking the importance of the features in its input. A set of weights was introduced to be used in the similarity function of EVREG inspired by the extension of the k-NN regression for feature selection. These weights are learned during the training phase, so that a set of weights which decreases the error of the original method can be found. The values of these weights represent the importance of each feature.

Initially, the method was tested in feature selection and prediction tasks using synthetic and well-known datasets. The results showed the ability of the model to solve these tasks in a competitive way when compared with other existing models. In particular, the method obtained better results for feature selection tasks with complex relations in the synthetic data experiments. Then we performed experiments using well-known datasets concluding that its performance is similar to other embedded methods. The prediction performance of the method was tested on the Boston housing, medical expenses, red wine quality and abalone datasets. The results are comparable to the most well-known regression algorithms such as ANN and GB.

An example of the additional information that the method provides for each prediction was shown in section 4.3. Since the proposed method is an embedded method, the importance of each feature can be obtained automatically. Furthermore, the outcome for an input can be explained using the mass of its k-nearest neighbors and the target variable domain because it uses a k-NN approach.

The proposed method was also evaluated with data taken from two real case scenarios. The first one was store traffic forecast, a time series forecasting problem. In this case, the number of visitors of 50 retail stores was predicted. The proposed method was compared against other methods which were tested and reported in the literature for this problem. The results showed that the proposed method is able to reach good prediction performance, obtaining the second place close behind RF. In this problem, confidence interval analysis was performed. Ideally the confidence interval of the target variable needs to be as sharp as possible while still having the real target inside. A new metric was introduced to measure these requirements. We then compared our method to others capable of giving a confidence interval in the response such as GP and SARIMA. Our method performed better in the confidence intervals of the response.

The second real case scenario for which data was available was in the realm of health care costs prediction. Here, three scenarios were tested where the available patient history varied. The first scenario used a single year of history to predict the costs of the next year, the second used 2 years to predict the third one and the last one used 5 years to predict the costs in the sixth year. In each scenario, the number of samples decreased while the number of features increased significantly. Initially, using all the features for each scenario the performance of the proposed method was one of the best ones obtaining second place behind GB, which is the best method tested in the literature for this problem. Regarding the feature selection performance, the proposed method obtained similar results to the other methods, but being able to have good performance even when most features were considered, a situation in which RT algorithm could not. The ranking of features obtained by the proposed method confirmed the suspicion of previous works suggesting the importance of previous costs features for the prediction of future costs. The results showed that previous costs alone are enough to predict future patients health care costs. An additional experiment was conducted where only the top 5 features were used. The results obtained were better than using all the available features.

Finally, the obtained results justify a claim that the proposed method is a valid regression method with comparable prediction performance to the most well-known regression methods. The method includes a ranking of features in its learning phase which makes it a valid embedded method. The performance in feature selection tasks is comparable to the best embedded methods. It is also a valid transparent method being capable of explaining each one of its predictions and give a confidence interval for each response unlike most of the methods that were tested in this thesis.

Future Work

In this work, a new transparent regression method was proposed. The method was able to obtain comparable results to well-known regression and embedded methods, but there is still room for improvements. During this work only one similarity function (RBF) with two distances measures were tested. New functions could be tested for specific problems that could yield to improvements in the method performance. For example, a neural network could be used to compute the similarity between two vectors although this would hurt the transparency of the method.

The computational complexity of a prediction method was explained in this work. Even though the time complexity for the prediction process grows linearly with the training set, the execution time of the algorithm needs to be further improved since it is longer than those of the benchmark methods. The execution time could be improved decreasing the complexity of the normalization term K computed in each prediction or by partitioning the space of the problem with a classification approach before each prediction.

The proposed method was tested only with unidimensional output. It would be interesting in a future study to work with a multidimensional one, for example comparing the feature selection process using a method for each dimension of the output versus a unique method predicting all outputs at once.

It is important to further investigate the proposed method interpretability. In this work the proposed method is fully transparent; this does not implies that it is always interpretable. The measure of a method interpretability is an open problem in the literature. There are interpretability measures for rule based algorithms, but this does not apply to the proposed method. An interpretability measure needs to be introduced to objectively measure the method interpretation. Another approach could be to measure interpretability in an experiment carried out with the help of experts for a specific problem as it was done by Peñafiel [114].

Contributions

The implementation of the proposed method can be obtained from `https://github.com/belisariops/WEVREG`, where there are also scripts for each one of the experiments presented in Chapter 4. Besides the implementation of the algorithm and this thesis, this work has also contributed to the articles listed below.

1. Panay, B., Baloian, N., Pino, J. A., Peñafiel, S., Sanson, H., & Bersano, N. (2019). Predicting Health Care Costs Using Evidence Regression. In Multidisciplinary Digital Publishing Institute Proceedings (Vol. 31, No. 1, p. 74).
2. Panay, B., Baloian, N., Pino, J. A., Peñafiel, S., Sanson, H., & Bersano, N. (2020). Feature Selection for Health Care Costs Prediction Using Weighted Evidential Regression. Sensors, 20(16), 4392.

Bibliography

[1] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[2] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[3] R. Wexler, "When a computer program keeps you in jail: How computers are harming criminal justice," *New York Times*, vol. 13, 2017.

[4] Z. Obermeyer, B. Powers, C. Vogeli, and S. Mullainathan, "Dissecting racial bias in an algorithm used to manage the health of populations," *Science*, vol. 366, no. 6464, pp. 447–453, 2019.

[5] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, 2015, pp. 2048–2057.

[6] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining.* ACM, 2016, pp. 1135–1144.

[7] J. Krause, A. Perer, and K. Ng, "Interacting with predictions: Visual inspection of black-box machine learning models," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems.* ACM, 2016, pp. 5686–5697.

[8] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[9] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018.

[10] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[11] P. R. Norvig and S. A. Intelligence, *A modern approach.* Prentice Hall, 2002.

[12] X. Hu, C. Rudin, and M. Seltzer, "Optimal sparse decision trees," in *Advances in Neural Information Processing Systems*, 2019, pp. 7267–7275.

[13] S. Peñafiel, N. Baloian, H. Sanson, and J. A. Pino, "Applying dempster–shafer theory for developing a flexible, accurate and interpretable classifier," *Expert Systems with Applications*, vol. 148, p. 113262, 2020.

[14] U. Johansson, H. Linusson, T. Löfström, and H. Boström, "Interpretable regression trees using conformal prediction," *Expert systems with applications*, vol. 97, pp. 394–404, 2018.

[15] S. Petit-Renaud and T. Denœux, "Nonparametric regression analysis of uncertain and imprecise data using belief functions," *International Journal of Approximate Reasoning*, vol. 35, no. 1, pp. 1–28, 2004.

[16] G. Shafer, "Dempster's rule of combination," *International Journal of Approximate Reasoning*, vol. 79, pp. 26–40, 2016.

[17] Z. Zhang, "Introduction to machine learning: k-nearest neighbors," *Annals of translational medicine*, vol. 4, no. 11, 2016.

[18] "Dempster's rule of combination," *Int. J. Approx. Reasoning*, vol. 79, no. C, pp. 26–40, Dec. 2016. [Online]. Available: https://doi.org/10.1016/j.ijar.2015.12.009

[19] M. Dash and H. Liu, "Feature selection for classification," *Intelligent data analysis*, vol. 1, no. 3, pp. 131–156, 1997.

[20] L. Yu and H. Liu, "Redundancy based feature selection for microarray data," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004, pp. 737–742.

[21] R. Ruiz, J. C. Riquelme, and J. S. Aguilar-Ruiz, "Incremental wrapper-based gene selection from microarray data for cancer classification," *Pattern Recognition*, vol. 39, no. 12, pp. 2383–2392, 2006.

[22] T. Hastie, R. Tibshirani, and J. Friedman, "The elements of statistical learning," 2017.

[23] J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, "Machine learning: a historical and methodological analysis," *AI Magazine*, vol. 4, no. 3, pp. 69–69, 1983.

[24] P. J. Brockwell and R. A. Davis, *Introduction to time series and forecasting.* springer, 2016.

[25] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016, http://www.deeplearningbook.org.

[26] M. T. Ribeiro, S. Singh, and C. Guestrin, "Model-agnostic interpretability of machine learning," *arXiv preprint arXiv:1606.05386*, 2016.

[27] T. Miller, “Explanation in artificial intelligence: Insights from the social sciences,” *Artificial Intelligence*, vol. 267, pp. 1–38, 2019.

[28] Z. C. Lipton, “The mythos of model interpretability,” *arXiv preprint arXiv:1606.03490*, 2016.

[29] C. Molnar, *Interpretable Machine Learning.* Lulu. com, 2020.

[30] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning.* Springer series in statistics New York, 2001, vol. 1, no. 10.

[31] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature extraction: foundations and applications.* Springer, 2008, vol. 207.

[32] J. Benesty, J. Chen, Y. Huang, and I. Cohen, “Pearson correlation coefficient,” in *Noise reduction in speech processing.* Springer, 2009, pp. 1–4.

[33] C. Goutte and E. Gaussier, “A probabilistic interpretation of precision, recall and f-score, with implication for evaluation,” in *European Conference on Information Retrieval.* Springer, 2005, pp. 345–359.

[34] P. Bugata and P. Drotár, “Weighted nearest neighbors feature selection,” *Knowledge-Based Systems*, vol. 163, pp. 749–761, 2019.

[35] G. Niu and B.-S. Yang, “Dempster–shafer regression for multi-step-ahead time-series prediction towards data-driven machinery prognosis,” *Mechanical systems and signal processing*, vol. 23, no. 3, pp. 740–751, 2009.

[36] P. Baraldi, F. Di Maio, S. Al-Dahidi, E. Zio, and F. Mangili, “Prediction of industrial equipment remaining useful life by fuzzy similarity and belief function theory,” *Expert Systems with Applications*, vol. 83, pp. 226–241, 2017.

[37] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice.* OTexts, 2018.

[38] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction.* Springer Science & Business Media, 2009.

[39] A. Navot, L. Shpigelman, N. Tishby, and E. Vaadia, “Nearest neighbor based feature selection for regression and its application to neural activity,” in *Advances in neural information processing systems*, 2006, pp. 996–1002.

[40] D. Basak, S. Pal, and D. C. Patranabis, “Support vector regression,” *Neural Information Processing-Letters and Reviews*, vol. 11, no. 10, pp. 203–224, 2007.

[41] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[42] V. Vapnik, *The nature of statistical learning theory.* Springer science & business media,

2013.

[43] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning.* MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[44] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[45] C. Lesmeister, *Mastering machine learning with R.* Packt Publishing Ltd, 2015.

[46] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in neurorobotics*, vol. 7, p. 21, 2013.

[47] K.-L. Du and M. Swamy, "Multilayer perceptrons: Architecture and error backpropagation," in *Neural Networks and Statistical Learning.* Springer, 2014, pp. 83–126.

[48] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[49] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[50] G. Shafer *et al.*, *A mathematical theory of evidence.* Princeton university press Princeton, 1976, vol. 1.

[51] S. E. Chick, "Subjective probability and bayesian methodology," *Handbooks in Operations Research and Management Science*, vol. 13, pp. 225–257, 2006.

[52] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "A review of feature selection methods on synthetic data," *Knowledge and information systems*, vol. 34, no. 3, pp. 483–519, 2013.

[53] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *International conference on database theory.* Springer, 2001, pp. 420–434.

[54] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[55] G. E. Fasshauer and J. G. Zhang, "On choosing "optimal" shape parameters for rbf approximation," *Numerical Algorithms*, vol. 45, no. 1-4, pp. 345–368, 2007.

[56] M. Mongillo, "Choosing basis functions and shape parameters for radial basis function methods," *SIAM undergraduate research online*, vol. 4, no. 190-209, pp. 2–6, 2011.

[57] P. Smets *et al.*, "What is dempster-shafer's model," *Advances in the Dempster-Shafer theory of evidence*, pp. 5–34, 1994.

[58] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, 2019.

[59] L. Devroye, "The uniform convergence of nearest neighbor regression function estimators and their application in optimization," *IEEE Transactions on Information Theory*, vol. 24, no. 2, pp. 142–151, 1978.

[60] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning," in *Lazy learning*. Springer, 1997, pp. 11–73.

[61] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[62] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[63] P. Rodoreda, "Feature selection with iterative feature weighting methods," Master's thesis, Universitat Politecnica de Catalunya, 1 2018.

[64] J. H. Friedman, "Multivariate adaptive regression splines," *The annals of statistics*, pp. 1–67, 1991.

[65] D. Freedman, R. Pisani, and R. Purves, "Statistics (international student edition)," *Pisani, R. Purves, 4th edn. WW Norton & Company, New York*, 2007.

[66] D. Harrison Jr and D. L. Rubinfeld, "Hedonic housing prices and the demand for clean air," 1978.

[67] B. Lantz, *Machine learning with R*. Packt Publishing Ltd, 2013.

[68] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Systems*, vol. 47, no. 4, pp. 547–553, 2009.

[69] S. G. Waugh, "Extending and benchmarking cascade-correlation: Extensions to the cascade-correlation architecture and benchmarking of feed-forward supervised artificial neural networks," Ph.D. dissertation, University of Tasmania, 1995.

[70] W. J. Reinartz and V. Kumar, "Store-, market-, and consumer-characteristics: The drivers of store performance," *Marketing Letters*, vol. 10, no. 1, pp. 5–23, 1999.

[71] I.-D. Anic, S. Radas, and L. K. Lim, "Relative effects of store traffic and customer traffic flow on shopper spending," *The International Review of Retail, Distribution and Consumer Research*, vol. 20, no. 2, pp. 237–250, 2010.

[72] O. Perdikaki, S. Kesavan, and J. M. Swaminathan, "Effect of traffic on sales and conversion rates of retail stores," *Manufacturing & Service Operations Management*, vol. 14, no. 1, pp. 145–162, 2012.

[73] S. Lam, M. Vandenbosch, and M. Pearce, "Retail sales force scheduling based on store traffic forecasting," *Journal of Retailing*, vol. 74, no. 1, pp. 61–88, 1998.

[74] S. Netessine, M. Fisher, and J. Krishnan, "Labor planning, execution, and retail store performance: An exploratory investigation," *Execution, and Retail Store Performance: An Exploratory Investigation (January 3, 2010)*, 2010.

[75] B. Veeling, "Improving visitor traffic forecasting in brick-and-mortar retail stores with neural networks," *Bachelor thesis, Universiteit Twente*, 2014.

[76] P. Cortez, L. M. Matos, P. J. Pereira, N. Santos, and D. Duque, "Forecasting store foot traffic using facial recognition, time series and support vector machines," in *International Joint Conference SOCO'16-CISIS'16-ICEUTE'16*. Springer, 2016, pp. 267–276.

[77] S. Abrishami, P. Kumar, and W. Nienaber, "Smart stores: A scalable foot traffic collection and prediction system," in *Industrial Conference on Data Mining*. Springer, 2017, pp. 107–121.

[78] B. M. Williams and L. A. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results," *Journal of transportation engineering*, vol. 129, no. 6, pp. 664–672, 2003.

[79] J. Guo, Y. Peng, X. Peng, Q. Chen, J. Yu, and Y. Dai, "Traffic forecasting for mobile networks with multiplicative seasonal arima models," in *2009 9th International Conference on Electronic Measurement & Instruments*. IEEE, 2009, pp. 3–377.

[80] S. V. Kumar and L. Vanajakshi, "Short-term traffic flow prediction using seasonal arima model with limited input data," *European Transport Research Review*, vol. 7, no. 3, p. 21, 2015.

[81] P. Cortez and M. J. Embrechts, "Using sensitivity analysis and visualization techniques to open black box data mining models," *Information Sciences*, vol. 225, pp. 1–17, 2013.

[82] S. Abrishami and P. Kumar, "Using real-world store data for foot traffic forecasting," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 1885–1890.

[83] V. Vovk, "On-line confidence machines are well-calibrated," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.* IEEE, 2002, pp. 187–196.

[84] W. H. Organization, "Public spending on health: a closer look at global trends," World Health Organization, Tech. Rep., 2018.

[85] A. M. Garber and J. Skinner, "Is american health care uniquely inefficient?" *Journal of Economic Perspectives*, vol. 22, no. 4, pp. 27–50, 2008.

[86] I. Yoo, P. Alafaireet, M. Marinov, K. Pena-Hernandez, R. Gopidi, J.-F. Chang, and L. Hua, "Data mining in healthcare and biomedicine: a survey of the literature," *Journal of medical systems*, vol. 36, no. 4, pp. 2431–2448, 2012.

[87] M. Bilger and W. G. Manning, “Measuring overfitting in nonlinear models: a new method and an application to health expenditures,” *Health economics*, vol. 24, no. 1, pp. 75–85, 2015.

[88] P. Diehr, D. Yanez, A. Ash, M. Hornbrook, and D. Lin, “Methods for analyzing health care utilization and costs,” *Annual review of public health*, vol. 20, no. 1, pp. 125–144, 1999.

[89] R. Kronick, T. Gilmer, T. Dreyfus, and T. Ganiats, “Cdps-medicare: The chronic illness and disability payment system modified to predict expenditures for medicare beneficiaries,” *Final Report to CMS*, 2002.

[90] M. A. Morid, K. Kawamoto, T. Ault, J. Dorius, and S. Abdelrahman, “Supervised learning methods for predicting healthcare costs: Systematic literature review and empirical evaluation,” in *AMIA Annual Symposium Proceedings*, vol. 2017. American Medical Informatics Association, 2017, p. 1312.

[91] D. Bertsimas, M. V. Bjarnadóttir, M. A. Kane, J. C. Kryder, R. Pandey, S. Vempala, and G. Wang, “Algorithmic prediction of health-care costs,” *Operations Research*, vol. 56, no. 6, pp. 1382–1392, 2008.

[92] S. Sushmita, S. Newman, J. Marquardt, P. Ram, V. Prasad, M. D. Cock, and A. Teredesai, “Population cost prediction on public healthcare datasets,” in *Proceedings of the 5th International Conference on Digital Health 2015*. ACM, 2015, pp. 87–94.

[93] I. Duncan, M. Loginov, and M. Ludkovski, “Testing alternative regression frameworks for predictive modeling of health care costs,” *North American Actuarial Journal*, vol. 20, no. 1, pp. 65–87, 2016.

[94] B. Mihaylova, A. Briggs, A. O’Hagan, and S. G. Thompson, “Review of statistical methods for analysing healthcare resources and costs,” *Health economics*, vol. 20, no. 8, pp. 897–916, 2011.

[95] D. K. Blough, C. W. Madden, and M. C. Hornbrook, “Modeling risk using generalized linear models,” *Journal of health economics*, vol. 18, no. 2, pp. 153–171, 1999.

[96] S. F. Leung and S. Yu, “On the choice between sample selection and two-part models,” *Journal of econometrics*, vol. 72, no. 1-2, pp. 197–229, 1996.

[97] A. H. Marshall, B. Shaw, and S. I. McClean, “Estimating the costs for a group of geriatric patients using the coxian phase-type distribution,” *Statistics in medicine*, vol. 26, no. 13, pp. 2716–2729, 2007.

[98] A. M. Jones *et al.*, *Models for health care.* University of York., Centre for Health Economics, 2009.

[99] S.-M. Lee, J.-O. Kang, and Y.-M. Suh, “Comparison of hospital charge prediction models for colorectal cancer patients: neural network vs. decision tree models,” *Journal of Korean medical science*, vol. 19, no. 5, pp. 677–681, 2004.

[100] E. W. Frees, X. Jin, and X. Lin, “Actuarial applications of multivariate two-part regression models,” *Annals of Actuarial Science*, vol. 7, no. 2, pp. 258–287, 2013.

[101] J. Elith, J. R. Leathwick, and T. Hastie, “A working guide to boosted regression trees,” *Journal of Animal Ecology*, vol. 77, no. 4, pp. 802–813, 2008.

[102] C. D. Sutton, “Classification and regression trees, bagging, and boosting,” *Handbook of statistics*, vol. 24, pp. 303–329, 2005.

[103] J. M. Zurada, *Introduction to artificial neural systems.* West publishing company St. Paul, 1992, vol. 8.

[104] L. Breiman, *Classification and regression trees.* Routledge, 2017.

[105] P. Tanuseputro, W. P. Wodchis, R. Fowler, P. Walker, Y. Q. Bai, S. E. Bronskill, and D. Manuel, “The health care cost of dying: a population-based retrospective cohort study of the last year of life in ontario, canada,” *PLoS one*, vol. 10, no. 3, p. e0121759, 2015.

[106] D. Howdon and N. Rice, “Health care expenditures, age, proximity to death and morbidity: Implications for an ageing population,” *Journal of Health Economics*, vol. 57, pp. 60–74, 2018.

[107] V. von Wyl, “Proximity to death and health care expenditure increase revisited: A 15-year panel analysis of elderly persons,” *Health economics review*, vol. 9, no. 1, p. 9, 2019.

[108] O. mondiale de la santé and W. H. Organization, *International classification of functioning, disability and health: ICF.* World Health Organization, 2001.

[109] S. Matsuda and K. Fujimori, “The claim database in japan,” *Asian Pacific Journal of Disease Management*, vol. 6, no. 3-4, pp. 55–59, 2014.

[110] D. Koller, G. Schön, I. Schäfer, G. Glaeske, H. van den Bussche, and H. Hansen, “Multimorbidity and long-term care dependency—a five-year follow-up,” *BMC geriatrics*, vol. 14, no. 1, p. 70, 2014.

[111] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning.* Springer, 2013, vol. 112.

[112] B. Jönsson, “Revealing the cost of type ii diabetes in europe,” *Diabetologia*, vol. 45, no. 1, pp. S5–S12, 2002.

[113] I. Köster, L. Von Ferber, P. Ihle, I. Schubert, and H. Hauner, “The cost burden of diabetes mellitus: the evidence from germany—the codim study,” *Diabetologia*, vol. 49, no. 7, pp. 1498–1504, 2006.

[114] S. Peñafiel, “Interpretable method for general classification using dempster shafer theory,” Master’s thesis, Universidad de Chile, 2020.

[115] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[116] S. Seabold and J. Perktold, "statsmodels: Econometric and statistical modeling with python," in *9th Python in Science Conference*, 2010.

[117] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Appendices

Appendix A

Methods Configuration

The configuration of the methods in the experiments shown in this thesis will be listed. For the methods implementation Scikt-learn [115], Statsmodels [116] and Pytorch [117] libraries were used. Scikit-Learn was used for GB, GP, SVR, RF, RT, ANN and KNN regressors. Statsmodels was used for its SARIMA implementation. And Pytorch was used for its implementation of tensor operations and gradient descent. Pytorch was also used for the implementation of the LSTM neural network.

A.1 Model evaluation

A.1.1 Synthetic data

The configurations of the methods used for the experiment with synthetic data are the following.

- **WEVREG2**:
  - Number of neighbors: 20
  - Maximum iterations: 50
  - Learning rate: 0.05
  - p: 2
- **WEVREG1**:
  - Number of neighbors: 20
  - Maximum iterations: 50
  - Learning rate: 0.05
  - p: 1
- **WkNN**:
  - Number of neighbors: 20
  - Maximum iterations: 50
  - Learning rate: 0.05
- **GB**:

  - Number of boosting stages: 100
  - Maximum number of nodes: 3
  - Learning rate: 0.1
- **RF**:
  - Number of estimators: 100
- **RT**:
  - Minimum samples for split: 2

A.1.2 Well-known data

The configuration of the experiments in the well-known regression datasets is detailed below.

- **WEVREG2**:
  - Number of neighbors: 30
  - Maximum iterations: 100
  - Learning rate: 0.1
  - p: 2
- **WEVREG1**:
  - Number of neighbors: 30
  - Maximum iterations: 100
  - Learning rate: 0.1
  - p: 1
- **WkNN**:
  - Number of neighbors: 30
  - Maximum iterations: 100
  - Learning rate: 0.1
- **GB**:
  - Number of boosting stages: 100
  - Maximum number of nodes: 3
  - Learning rate: 0.1
- **SVR**:
  - C: 1,000
  - Kernel: Linear
  - Epsilon: 0.1
- **RF**:
  - Number of estimators: 100
- **RT**:
  - Minimum samples for split: 2
- **ANN**:
  - Hidden layer sizes: (50, 25, 10)

  - Learning rate: 0.1
  - Maximum iterations: 800
  - Activation: Rectified Linear Unit
- **KNN**:
  - Number of neighbors: 30

A.2 Store Traffic

The configuration used for every method in the store traffic forecasting problem is detailed below:

1. **WEVREG**
   - Number of neighbors: 15
   - Maximum iterations: 50
   - Learning rate: 0.1
2. **RF**
   - Number of estimators: 350
   - Minimum samples split: 2
   - Number of features considered for split: 12
3. **SVR**
   - C: 100
   - Epsilon: 0.1
   - Kernel: RBF
4. **GP**
   - Kernel: Matérn
5. **LSTM**
   - Hidden layer size: 128
   - Layers: 1
   - Learning rate: 0.01
   - Maximum iterations: 250
6. **SARIMA**
   - (p, d, q): $(1, 1, 0)$
   - $(P, D, Q)_s$: $(1, 0, 0)_7$

A.3 Health Care Costs

The configuration of every method in each scenario of the health care cost prediction problem presented is detailed below.

Scenario 1

1. **WEVREG**
    - Number of neighbors: 20
    - Maximum iterations: 20
    - Learning rate: 0.1
2. **RT**
    - Minimum number of samples required to split: 950
    - Number of features consider for a split: 3
    - Maximum depth: 30
3. **WkNN**
    - Number of neighbors: 20
    - Maximum iterations: 30
    - Learning rate: 0.01
4. **ANN**
    - Hidden layer sizes: (152, 10)
    - Learning rate: 0.1
    - Maximum iterations: 800
    - Activation: Rectified Linear Unit
5. **GB**
    - Maximum depth of individual estimator: 1
    - Number of features consider for a split: 2
    - Number of boosting stages: 125

Scenario 2

1. **WEVREG**
    - Number of neighbors: 20
    - Maximum iterations: 20
    - Learning rate: 0.1
2. **RT**
    - Number of estimators: 350
    - Minimum samples split: 2
    - Number of features considered for split: 12
3. **WkNN**
    - Number of neighbors: 20
    - Maximum iterations: 30
    - Learning rate: 0.01
4. **ANN**
    - Hidden layer sizes: (308, 10)

- Learning rate: 0.1
- Maximum iterations: 800
- Activation: Rectified Linear Unit

5. **GB**
   - Maximum depth of individual estimator: 1
   - Number of features consider for a split: 2
   - Number of boosting stages: 125

Scenario 3

1. **WEVREG**
   - Number of neighbors: 20
   - Maximum iterations: 20
   - Learning rate: 0.1
2. **RT**
   - Minimum number of samples required to split: $1,000$
   - Number of features consider for a split: 3
   - Maximum depth: 30
3. **WkNN**
   - Number of neighbors: 20
   - Maximum iterations: 30
   - Learning rate: 0.01
4. **ANN**
   - Hidden layer sizes: (764, 10)
   - Learning rate: 0.1
   - Maximum iterations: 800
   - Activation: Rectified Linear Unit
5. **GB**
   - Maximum depth of individual estimator: 1
   - Number of features consider for a split: 2
   - Number of boosting stages: 125