1 2 9 0

UNIVERSIDADE Đ COIMBRA

Frederico Carvalho Álvares de Melo Ferreira

MACHINE LEARNING TECHNIQUES IN CONNECTION DESIGN

Dissertation in the context of the Master in Steel and Composite Construction, advised by Prof. Trayana Tankova and Prof. Catarina Silva and presented to the Faculty of Sciences and Technology / Department of Civil Engineering

October 2021

This page is intentionally left blank.

Faculdade de Ciências e Tecnologia da Universidade de Coimbra
Departamento de Engenharia Civil

Frederico Carvalho Álvares de Melo Ferreira

MACHINE LEARNING TECHNIQUES IN CONNECTION DESIGN

Dissertação de Mestrado em Construção Metálica e Mista,
orientada pela Professor Doutora Trayana Tankova e pela Professora Doutora Catarina Silva

Dezembro de 2021

1 2 9 0

UNIVERSIDADE Ð
COIMBRA

This page is intentionally left blank.

Acknowledgements

This work was only possible thanks to the support, patience and friendship of my family, friends and colleagues.

Special acknowledgments have to be made:

- To Professor Trayana Tankova and Catarina Silva for their support, knowledge and never ending patience which were essential for the development of the current work.

- To my friends António Castilho, Diogo Rodrigues, Filipe Valente, Gonçalo Ferreira Miguel Pereira and Pedro Pereira for their friendship and support throughout this long period.

- To my friends Bruno Evangelista, Francisco José, José José, Miguel Braz and Pedro Horta for their support as well as for all the great moments which gave me the strength to work even harder.

- To Afaconsult and all my colleagues for giving me the chance not only to further develop my Engineering skills but also to be part of exciting and challenging Engineering projects.

- To Ana Semedo for her invaluable care, encouragement and support without which this work would have never been possible.

- To my Parents, for everything. Without their incredible support and patience it would have never been possible for me to get to where I am today.

This page is intentionally left blank.

Abstract

The use of the computer has become an essential part in the work environment of companies across a wide range of areas of expertise. The increase in computational power throughout the years led to faster analysis with increased quality, enabled the transfer of information between the different stakeholders in an easier and intelligible manner while also allowing the use of computational learning techniques which are applied to the vast amount of data gathered during the last decades, leading to significant developments in different areas such as publicity, economics, medicine, robotics, just to name a few.

Despite the fast growth of this digitalization trend observed in other areas, Civil Engineering is just now starting to implement more advanced techniques that may be someday considered a reliable alternative to the classic approaches.
The design process is currently based upon empirical knowledge, advanced analytical methods, experimental tests and numerical simulations, used as the basis for the design of different structural elements, and which despite being of fairly easy implementation constitute time consuming processes.

This work explores the potential of applying machine learning techniques for the design of steel connections and the prediction of their corresponding failure modes, without the need to compute the expressions proposed by EN1993-1-8, by using different learning algorithm in order to find the one that best fits the problem.

The procedure included in the European standard EN1993-1-8 for the design of welded steel connections and the equations that return the resistance of the different components of these connections are presented as well as the main characteristics of the various considered learning algorithms, the different metrics used to evaluate the quality of the created models.

An overall view of the workflow used to create both the dataset and the different models is also included

Finally, an analysis of the obtained results for both the classification of the conditioning component and the regression of the resistant bending moment is presented.

This page is intentionally left blank.

Resumo

O uso do computador tornou-se uma parte essencial no ambiente de trabalho de empresas que atuam nas mais diversas áreas. O aumento da capacidade computacional ao longo do tempo levou a análises mais rápidas e de melhor qualidade, à transmissão de informação entre os vários intervenientes de uma forma mais fácil e clara, permitindo ainda a utilização de técnicas de aprendizagem automática aplicadas à grande quantidade de dados gerados nas últimas décadas, que possibilitaram avanços significativos em áreas tão distintas como a publicidade, economia, medicina, robótica entre outros.

Apesar do rápido crescimento desta tendência de digitalização verificada noutras áreas, a Engenharia Civil começa apenas agora a implementar técnicas mais avançadas que possam eventualmente servir de alternativa às abordagens clássicas de projeto e execução, onde os procedimentos de dimensionamento são atualmente baseados na experiência, em métodos analíticos avançados, ensaios experimentais e simulações numéricas, que servem de base ao dimensionamento dos diferentes elementos estruturais, e que apesar de serem de fácil implementação, constituem em certos casos processos demorados.

No trabalho aqui desenvolvido, explora-se o potencial de aplicação da aprendizagem automática ao processo de dimensionamento de ligações e a possibilidade de previsão do diferentes modos de falha sem a necessidade de calcular as expressões fornecidas pela EN1993-1-8, recorrendo-se para tal a diferentes abordagens de forma a permitir encontrar aquela que melhor se adapta ao problema.

É feita uma apresentação do procedimento incluído na norma Europeia EN1993-1-8 para o dimensionamento de ligações soldadas e das respetivas equações destinadas à determinação da resistência das diferentes componentes são apresentadas. São ainda descritas as principais características dos diferentes algoritmos de aprendizagem e as principais medidas usadas para a avaliação da qualidade dos modelos criados.

É igualmente incluída uma perspetiva geral do procedimento usado para criar não só a base de dados mas também os diferentes modelos.

Por último é realizada uma análise dos resultados obtidos para os dois tipos de problemas considerados, o problema de classificação da componente condicionante e o problema de regressão do momento fletor resistente.

This page is intentionally left blank.

CONTENTS

LIST OF FIGURES

This page is intentionally left blank.

LIST OF TABLES

1 INTRODUCTION

1.1 Motivation

The use of steel elements as a structural solution for buildings, intended for a wide range of purposes, has become a usual option for structural engineering practitioners worldwide. This can be seen as a result of the different advantages that steel can offer, such as its reduced production costs when compared to other construction materials, the speed of the construction process which enables globally more economical solutions, the high resistance of steel, leading to lighter and slender structural elements and thus to more aesthetically appealing solutions, the combination of steel with other materials such as concrete, leading to composite solutions which make use of the best mechanical characteristics of each material.

The design of steel structures leads, invariably, to the design of the respective joints, regardless of their configuration (welded or bolted connections). Nowadays, the design of steel joints constitutes a fairly simple process, owing to the introduction of the computer as an essential part of the process of structural design among practitioners. However, despite the exponential increase in computational power witnessed throughout the last few years and the accompanying evolution of steel design software with an analytical based approach or by means of the Finite Element Method (FEM), the steel connection design process still constitutes a time consuming task, one that is painstakingly repeated despite the existing similarities between the elements to be connected and the forces involved, leading to steel connections with similar geometrical configurations not only within the same project but also across different projects. The design of steel connections is therefore a process for which there is a large amount of previously existing and available data within organizations such as a structural engineering practitioner, data that is generated and validated numerous times following older engineering projects and that most of the times is not reused as a basis for the design of new steel joints. Thus, this design process presents itself as a possible candidate for the application of machine learning techniques that could potentially lead to a more expeditious procedure, leading to lower time consumptions during the steel connection design process throughout the different stages of a project, contributing for the reduction of the associated costs.

1.2 Contributions

The construction of an exhaustive database of unreinforced welded beam to column connections is a crucial contribution for the main objective of this work. This database is comprised of different pairs of standard commercial profiles, and in particular those used in Europe such as IPE, HEA and HEB, or profiles based upon the combination of these, and their respective dimensional variations usually measured. Once obtained, the complete database can then be split into training and testing sets, the former used to build the models by training a learning algorithm and the latter to evaluate and validate the said model.

1.3 Objectives

The main objective of this thesis is the application of different learning algorithms to data in order to solve two different problems. The first consists in building models that allow the prediction of the conditioning component of the steel connections, in what is known as a classification problem, while the second is the prediction of the corresponding design resistant bending moment by solving a regression task.

1.4 Thesis Organization

This document is divided into six chapters.

The first chapter contains a brief introduction in which some of the advantages for the use of steel as a construction material are laid out. Moreover, the mainstream approaches followed for the design of steel connections within a practitioner's environment are also presented.

The procedure followed by the European standard EN 1993-1-8, for the design of welded beam to column steel connections, known as the Component Method, is presented in detail throughout Chapter 2.

Chapter 3 begins with a short presentation of Artificial Intelligence (AI) in general and Machine Learning (ML) in particular. The general aspects, which are transversal to different algorithms, are also presented, including the different types of learning in which these algorithms are inserted, the different types of tasks, as well as the metrics usually used to evaluate an algorithm's performance. Chapter 3 ends with the presentation of the basic concepts underpinning the most relevant algorithms.

The definition of the problem being investigated, the proposed approach as well as the procedure followed to obtain the dataset are presented along Chapter 4, while the obtained results, for both the classification and regression tasks are presented and discussed throughout Chapter 5.

Chapter 6 contains the resulting conclusions as well as future developments that may be pursued based upon the current work.

2 DESIGN OF WELDED BEAM TO COLUMN CONNECTIONS

2.1 Introduction

The design of steel connections implies their characterization in terms of resistance, stiffness and ductility. This can be achieved by means of experimental tests, advanced numerical analyse using the FEM and/or by analytical expressions.

Taking into account the considerable number of variables involved, the need to reduce the costs and optimize the time consumed during the design process of the different structural elements, expeditious methods, which enable the designer to engage in a process where iterations can be developed in a quick and simple manner, should be adopted. Thus, the analytical approach presents itself as the most practical one for the design if steel connections in the engineering practice.

The component method is the analytical method of choice for the design of steel connections with a wide range of geometrical configurations, intended to connect steel elements with different cross-sections subjected to various loads and load combinations, thus being adopted as the base procedure in the Eurocodes for the safety verification of this connections (Jaspart, J. P., Weynand, K. (2016)).

2.2 Component Method

The application of the component method to the design of connections is based upon the assumption that these connections are made up of a set of individual components, hereinafter referred to as basic components. The basic components may be subjected to stresses of various natures such as tension, compression and shear stresses as well as to an interaction of different stresses.

For each one of the basic components, their respective resistance and stiffness may be determined, allowing for the analysis of the global behaviour of the joint.
Taking into account the need to divide the joint in its respective basic components, the design of structural connections can be divided into different steps, which for the particular case of welded connections are as follows:

- Determination of the distribution of forces in both flanges of the beam, considering the acting axial force and bending moment.
- Determination of the resistance and safety assessment in the tension zone.

- Determination of the resistance and safety assessment in the compression zone.
- Determination of the resistance and safety assessment of the column's web in shear.
- Determination of the resistance and safety assessment of the welds to the flanges and web.

2.2.1 Force Distribution in the Beam

The tension $F_{t,Ed}$ and compression $F_{c,Ed}$ design forces acting on the beam flanges can be determined assuming that the contribution of the beam web to the transmission of both the bending moments and the axial forces to the column is negligible. Hence, the design forces acting on the beam flanges are exclusively a function of the distance between the centre of gravity of the beam flanges $(h_b - t_{fb})$.

$$F_{t,Ed} = \frac{M_{Ed}}{(h_b - t_{fb})} - \frac{N_{Ed}}{2} \tag{2.1}$$

$$F_{c,Ed} = \frac{M_{Ed}}{(h_b - t_{fb})} + \frac{N_{Ed}}{2} \tag{2.2}$$

where h_b is the beam height and t_{fb} is the thickness of the beam flange.

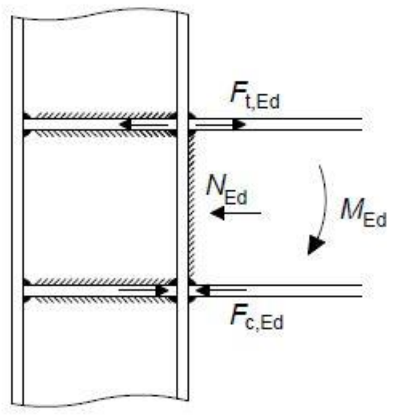


Figure 2.1 – Force distribution (SCI/BCSA. (2013))

2.2.2 Column Flange in Bending

The resistance of the column flange in bending connected to an unstiffened column depends upon the dispersion of the tension force between the beam flange and the column web. This dispersion can be taken into account by the effective width b_{eff}.

$$b_{eff} = t_{wc} + 2s + 7kt_{fc} \tag{2.3}$$

where t_{wc} is the thickness of the column web, s is the root radius of the column for rolled I or H sections or $s = \sqrt{2}a$ (a being the weld throat) for welded I or H sections, t_{fc} is the thickness of the column flanges and $k = \left(\frac{t_{fc}}{t_{fb}}\right)\left(\frac{f_{y,c}}{f_{y,b}}\right) \leq 1$ (4) with $f_{y,c}$ and $f_{y,b}$ being the design yield strength of the column and the beam respectively.
The effective width is limited by both the beam ($b_{eff} \leq b_b$) and column ($b_{eff} \leq b_c$) width.

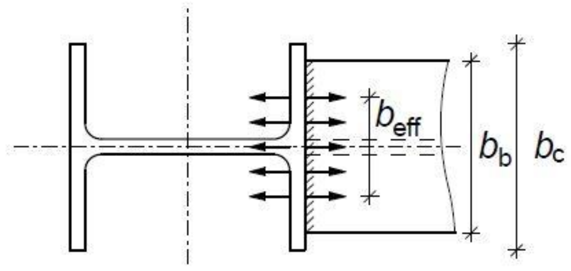


Figure 2.2 – Beam tension flange effective width (SCI/BCSA. (2013))

The design value for the resistance of the effective width of the beam flange is given by

$$F_{t,fb,Rd} = \frac{b_{eff} t_{fb} f_{y,fb}}{\gamma_{M0}} \tag{2.5}$$

2.2.3 Column Web in Tension

Analogous to the beam tension flange, the resistance of the column web in tension depends on the dispersion of the tension force between the two elements. In this case however the dispersion can be taken with a ratio of 1/2.5 of both the root radius and column flange thickness, limited in any case by the end of the column, and thus, the effective length of the column web is given by expression (2.6).

$$b_{eff,t,wc} = t_{fb} + 2s_f + 5(s + t_{fc}) \tag{2.6}$$

$s_f = \sqrt{2}a$ being the flange weld length.

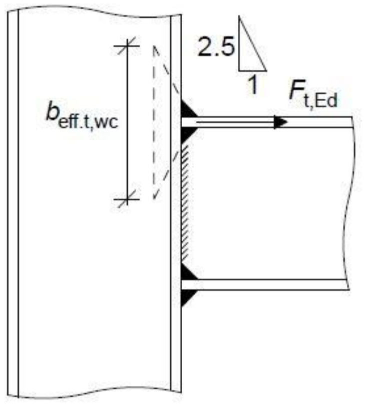


Figure 2.3 – Column web in tension effective length (SCI/BCSA. (2013))

The design resistance of the column web component can be determined from expression (2.7)

$$F_{t,wc,Rd} = \frac{\omega b_{eff,t,wc} t_{wc} f_{y,wc}}{\gamma_{M0}} \tag{2.7}$$

In order to take into account the interaction between the tension and shear stresses acting on the column web, expression (2.7) includes the reduction factor ω which in its turn depends upon the transformation parameter β.

Transformation parameter β	Reduction factor ω
$0 \leq \beta \leq 0.5$	$\omega = 1$
$0.5 < \beta < 1$	$\omega = \omega_1 + 2(1-\beta)(1-\omega_1)$
$\beta = 1$	$\omega = \omega_1$
$1 < \beta < 2$	$\omega = \omega_1 + (\beta - 1)(\omega_2 - \omega_1)$
$\beta = 2$	$\omega = \omega_2$
$\omega_1 = \frac{1}{\sqrt{1 + 1.3(b_{eff,c,wc} t_{wc}/A_{vc})^2}}$	$\omega_2 = \frac{1}{\sqrt{1 + 5.2(b_{eff,c,wc} t_{wc}/A_{vc})^2}}$
A_{vc} is the shear área of the column B is the transformation parameter	

Figure 2.4 – Reduction factor for interaction with shear (adapted from Comite Europeen de Normalisation. (2010))

The transformation parameter β allows for the relation between the shear force developed on the web panel and the compressive and tensile connection forces. Since the value of β is a function of the internal forces acting on the joint, and these forces for their turn variate with the stiffness of the joint, an iterative process would be required. Such a process can be avoided for practical application as long as adequate values of β are considered and used to model the joints beforehand, leading to a safe non-iterative analysis of the structure.

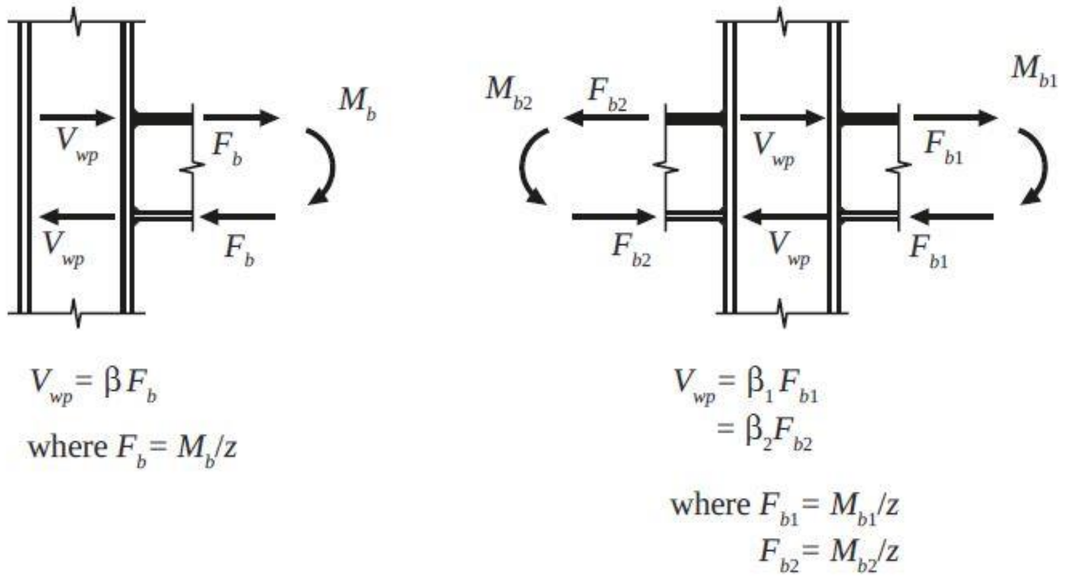


Figure 2.5 – Transformation parameter β (Jaspart, J. P., Weynand, K. (2016))

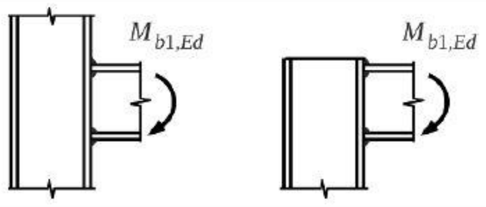
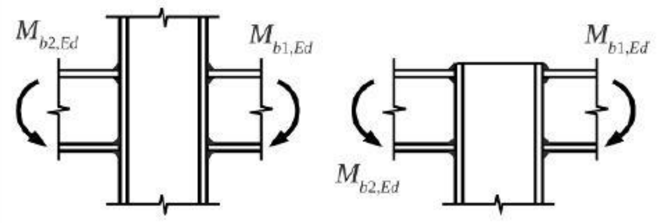
Type of joint configuration	Action	Value
	$M_{b1,Ed}$	$\beta \approx 1$
	$M_{b1,Ed} = M_{b2,Ed}$	$\beta = 0$ *)
	$M_{b1,Ed}/M_{b2,Ed} > 0$	$\beta \approx 1$
	$M_{b1,Ed}/M_{b2,Ed} < 0$	$\beta \approx 2$
	$M_{b1,Ed} + M_{b2,Ed} = 0$	$\beta \approx 2$
*) In this case the value of β is the exact value rather than an approximation		

Figure 2.6 – Transformation parameter β approximate values (Jaspart, J. P., Weynand, K. (2016))

If the resistance of the beam flange and column web in the tension zone are adequate, column tension stiffeners may be disregarded, which can be expressed as,

$$F_{t,Ed} \leq F_{t,fb,Rd} \tag{2.8}$$

$$F_{t,Ed} \leq F_{t,wc,Rd} \tag{2.9}$$

If either one of expressions (2.8) or (2.9) or $b_{eff} < \left(\frac{f_{y,b}}{f_{u,b}}\right) b_b$ are not verified, a pair of partial or full depth stiffeners should be provided.

These stiffeners contribute simultaneously to the tension resistance of the column web as well as to restrict the bending of the column's flange and their minimum area should be determined to ensure the design force $F_{s,Ed}$.

$$F_{s,Ed} = F_{t,fb,Rd} - \frac{\omega b_{eff,t,wc} t_{wc} f_{y,c}}{\gamma_{M0}} \tag{2.10}$$

The recommended overall width of each stiffener should be,

$$b_{sg} \geq \frac{0.75(b_c - t_{wc})}{2} \tag{2.11}$$

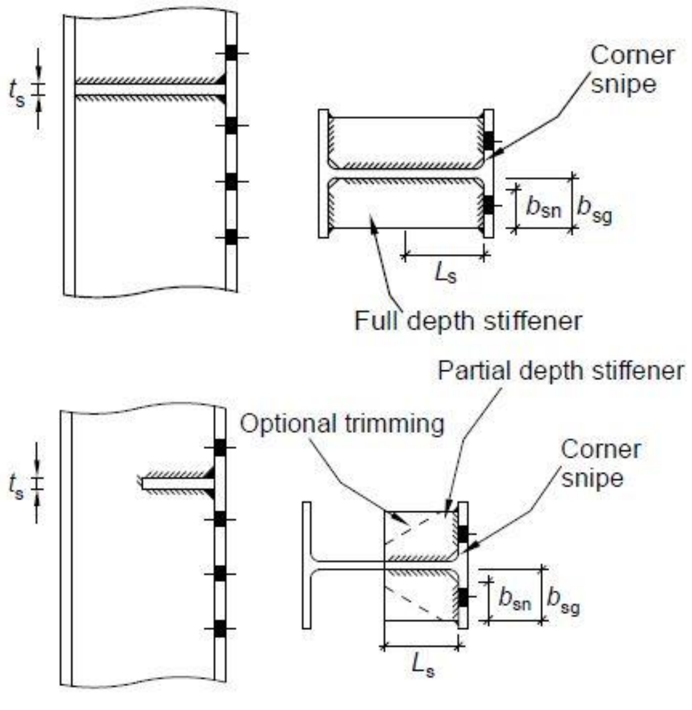


Figure 2.7 – Tension stiffeners (SCI/BCSA. (2013))

2.2.4 Beam Flange in Compression

The effective width b_{eff} of the beam flange in the compression zone is as given for the tension zone.

The resistance of the beam compression flange connected to an unstiffened column is given by,

$$F_{c,fb,Rd} = \frac{b_{eff} t_{fb} f_{y,fb}}{\gamma_{M0}} \tag{2.12}$$

2.2.5 Column Web in Compression

The resistance of the column web in the compression zone $F_{c,wc,Rd}$ depends, as was the case for the web in the tension zone, upon the dispersion of the compression force through the end plate, the column flange and the root radius, leading to

$$b_{eff,c,wc} = t_{fb} + 2s_f + 5(s + t_{fc}) + s_p \tag{2.13}$$

where $s_p = 2t_p$ when an end plate of t_p thickness is provided.

$$F_{c,wc,Rd} = \frac{\omega k_{wc} b_{eff,c,wc} t_{wc} f_{y,wc}}{\gamma_{M0}} \tag{2.14}$$

$$F_{c,wc,Rd} \leq \frac{\omega k_{wc} \rho b_{eff,c,wc} t_{wc} f_{y,wc}}{\gamma_{M1}} \tag{2.15}$$

Reduction coefficient k_{wc} allows for coexisting longitudinal compressive stress $\sigma_{com,Ed}$ in the column and is given by,

$$\sigma_{com,Ed} \leq 0.7 f_{y,wc} \quad k_{wc} = 1.0 \tag{2.16}$$

$$\sigma_{com,Ed} > 0.7 f_{y,wc} \quad k_{wc} = 1.7 - \frac{\sigma_{y,wc}}{f_{y,wc}} \tag{2.17}$$

Where $\sigma_{y,wc}$ is the maximum compressive stress acting on the column's web.
For the cases in which the column is in tension throughout, $k_{wc} = 1.0$.

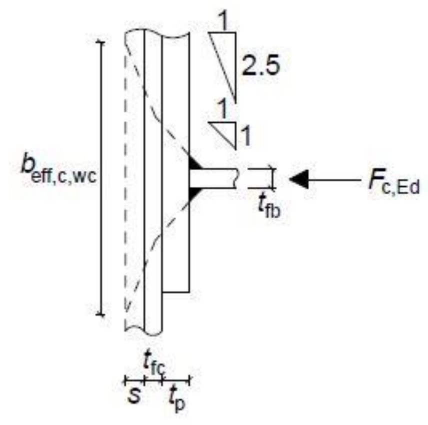


Figure 2.8 – Column web in compression effective length (SCI/BCSA. (2013))

The reduction factor ρ in expression (2.15) allows for the consideration of plate buckling and is given by,

$$\bar{\lambda}_p \leq 0.72 \qquad \rho = 1.0 \tag{2.18}$$

$$\bar{\lambda}_p > 0.72 \qquad \rho = \frac{\bar{\lambda}_p - 0.2}{\bar{\lambda}_p{}^2} \tag{2.19}$$

Where,

$$\bar{\lambda}_p = 0.932\sqrt{\frac{b_{eff,c,wc}d_{wc}f_{y,wc}}{Et_{wc}{}^2}} \tag{2.20}$$

and

$$d_{wc} = h_c - 2(t_{fc} + s) \tag{2.21}$$

If both the resistance of the flange and column web in the compression zone are adequate, compression stiffeners may be disregarded, and both expressions (2.22) and (2.23) are verified.

$$F_{c,Ed} \leq F_{c,fb,Rd} \tag{2.22}$$

$$F_{c,Ed} \leq F_{c,wc,Rd} \tag{2.23}$$

If either one of expressions (2.22), (2.23) or $b_{eff} < \left(\frac{f_{y,b}}{f_{u,b}}\right) b_b$ are not verified then full depth compression stiffeners are required and should be provided symmetrically on both sides of the column web. Partial depth stiffeners are also allowed although their application demands a more complex analysis of web buckling phenomena.

In order for a stiffened column to be considered to be appropriately designed, both the resistance of the effective stiffener cross section as well as its buckling resistance should be at least equal to the design force acting at the compression flange.

The effective section of the compression stiffener can be determined considering a cruciform cross section comprised of a $15\varepsilon t_{wc}$ length of web at either side of the added stiffener and a width b_{sg} of the later that depending on its thickness t_s should be limited to,

$$b_{sg} \leq 14 s t_s \tag{2.24}$$

in order to avoid torsional buckling, which corresponds to complying with the Class 3 limit for compression flange outstands.

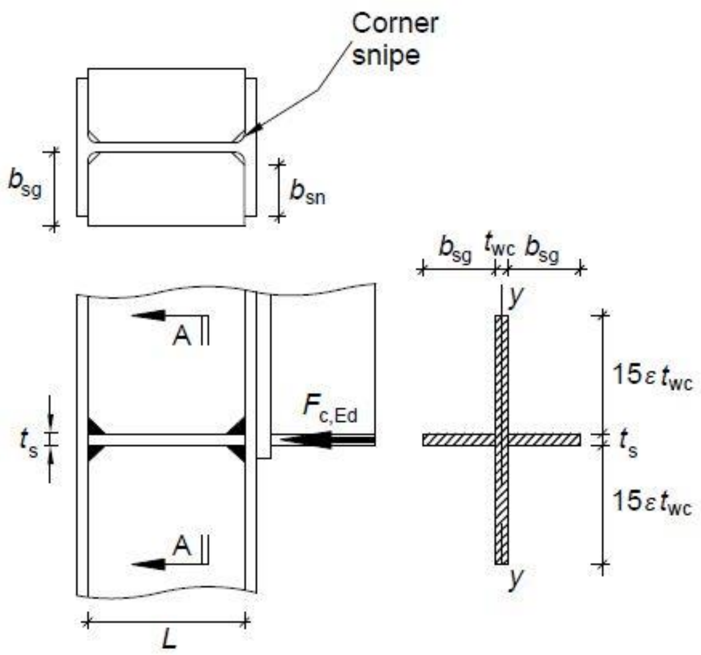


Figure 2.9 – Compression stiffeners (SCI/BCSA. (2013))

The buckling resistance can be determined by taking into account the effective area $A_{s,eff}$ and the second moment of area of the stiffener I_s

$$A_{s,eff} = (30 s t_{wc} + t_s) t_{wc} + 2 b_{sg} t_s \tag{2.25}$$

$$I_s = \frac{(2b_{sg} + t_{wc})^3 t_s}{12} \tag{2.26}$$

The above mentioned cross sectional resistance of the effective compression stiffener can be determined by expressin (2.27),

$$N_{c,Rd} = \frac{A_{s,eff} f_y}{\gamma_{M0}} \tag{2.27}$$

The stiffener's flexural buckling resistance depends as it is the case for other elements, upon their respective non-dimensional slenderness, which for the case of the compression stiffener may be obtained through expression (2.28),

$$\bar{\lambda} = \frac{l}{i_s \lambda_1} \tag{2.28}$$

Where $\lambda_1 = 93.9\varepsilon$, the radius of gyration $i_s = \sqrt{I_s/A_{s,eff}}$ and l is the critical buckling length of the stiffener, which for columns restrained against twist may be assumed to be not less than $l = 0.75h_w$ and for the cases where the column has no restrains against twist should be considered to be $l = h_w$.

For the cases in which $\bar{\lambda} \leq 0.2$, the stiffener's flexural buckling may be disregarded and only the cross sectional resistance should be verified.

If $\bar{\lambda} > 0.2$, the flexural buckling resistance can be determined by,

$$N_{b,Rd} = \frac{\chi A_{s,eff} f_y}{\gamma_{M1}} \tag{2.29}$$

Where,

$$\chi = \frac{1}{\Phi + \sqrt{\Phi^2 - \bar{\lambda}^2}} \leq 1 \tag{2.30}$$

$$\Phi = 0.5\left(1 + \alpha\left(\bar{\lambda} - 0.2\right) + \bar{\lambda}^2\right) \tag{2.31}$$

with $\alpha = 0.49$ and f_y is the minimum yield strength of either the column or the stiffener.

2.2.6 Column Web Panel Shear

The shear resistance of the column web panel may be the conditioning basic component of the connection for the cases of double-sided connections with either unbalanced or same direction bending moments and for single-sided beam to column connections.

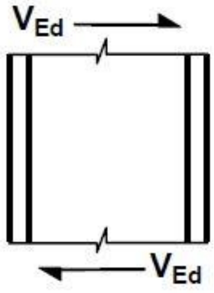


Figure 2.10 – Column web panel shear (Comite Europeen de Normalisation. (2010))

The shear force acting in the column web panel $V_{wp,Ed}$ of a single-sided connection may be conservatively taken as equal to the compression force $F_{c,Ed}$ for the cases where no axial force is acting on the beam. If there are axial forces acting on the beam to be connected, their effect should be considered.
For two-sided connections, the shear force value will depend upon the direction of the bending moment. For the cases in which moments with opposing directions arise, $V_{wp,Ed} = |F_{c,Ed,1} - F_{c,Ed,2}|$, otherwise $V_{wp,Ed} = F_{c,Ed,1} + F_{c,Ed,2}$.

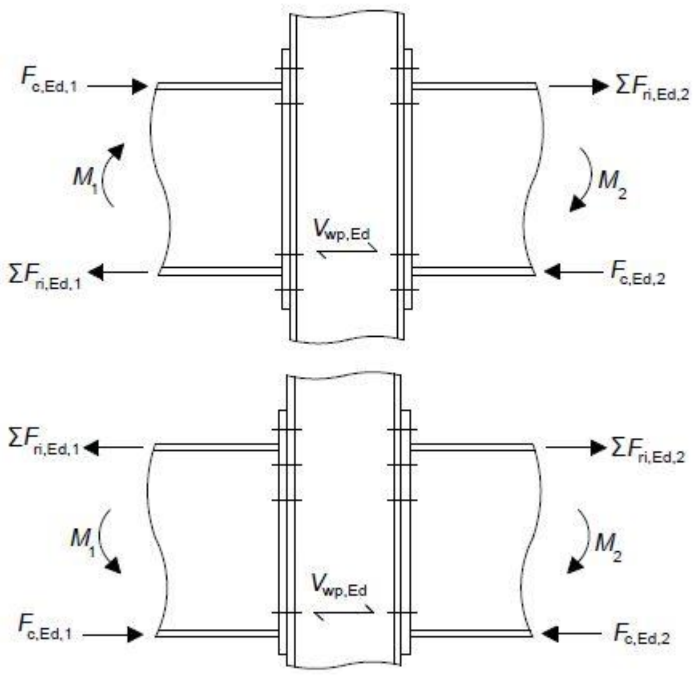


Figure 2.11 – Shear force design force (SCI/BCSA. (2013))

The shear resistance of the column web panel, for an unstiffened web with a slenderness limited to $d_c/t_{wc} \leq 69\varepsilon$ may be determined as,

$$V_{wp,Rd} = \frac{0.9 f_{yc} A_{vc}}{\gamma_{M0}\sqrt{3}} \tag{2.32}$$

where, $d_c = h_c - 2(t_{fc} + s)$ is the clear height of the column web, A_{vc} is the column's shear area and f_{yc} is the column's yield strength.

For the cases in which $d_c/t_{wc} > 69\varepsilon$ nothing is mentioned in EN 1993-1-8, being suggested however that,

$$V_{wp,Rd} = 0.9 V_{bw,Rd} \tag{2.33}$$

$V_{bw,Rd}$ is the web's contribution for the shear resistance of reinforced or unreinforced webs as defined by EN 1993-1-5 and may be obtained from,

$$V_{bw,Rd} = \frac{\chi_w f_{yw} h_w t}{\sqrt{3}\gamma_{M1}} \tag{2.34}$$

where χ_w is the factor for the contribution of the web to the shear buckling resistance.

For the cases for which the web resistance is not sufficient on its own, supplementary web plates or diagonal stiffeners may be considered.

When the web's reinforcement is achieved by means of supplementary web plates, additional requirements must be fulfilled:

- The same steel grade should be considered for both the column and the supplementary plates.
- The thickness of the supplementary web plates should be at least that of the column web panel.
- The width of the supplementary web plates should extend to the fillets of the column while not exceeding $40st_s$.
- The length of the supplementary web plate should extend at least to the effective length of the tension and compression zones of the column web.

Whenever these requirements are fulfilled, the supplementary web plate contributes for the increase in the joint resistance:

- 50% of web tension resistance for an additional plate on one side or a 100% if two additional plates, one on each, side are considered.
- 50% of web compression resistance for an additional plate on one side or a 100% if two additional plates, one on each, side are considered.
- 75% of web panel shear resistance for either one or two additional plates.

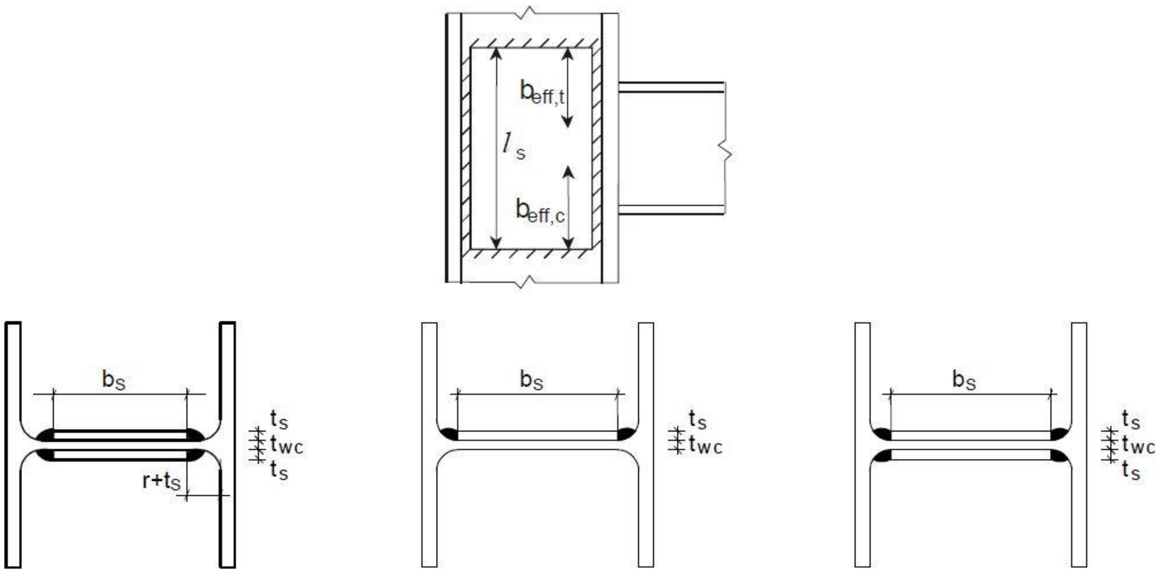


Figure 2.12 – Supplementary web plates dimensions (SCI/BCSA. (2013))

The contribution of supplementary web plates for the area of a column web panel is equal to $b_s t_{wc}$. Since that the increase in resistance is independent of the thickness of the additional plates, only one plate will contribute to the column web panel shear resistance. Furthermore, for the cases for which the supplementary plates are only required for the increase of shear resistance, the width of the plates is only required to be such that the toes of the fillet welds reach the fillets of the column section.

If the supplementary web plates are used as reinforcement for the tension resistance, their contribution depends on the throat thickness of the welds that connect them to the web and the effective thickness of the column web plate may be assumed to be $t_{w,eff} = 1.5t_{wc}$ or $t_{w,eff} = 2t_{wc}$ respectively for one or two supplementary plates.

The increase of the column web compression resistance, achieved through supplementary web plates, can be determined in an analogous way as for the case of tension resistance, considering an effective thickness of $t_{w,eff} = 1.5t_{wc}$ or $t_{w,eff} = 2t_{wc}$ respectively for one or two supplementary plates and the reduction factor ω in expression (2.14) and (2.15) should be based on the increased shear area.

In the cases where the web's reinforcement is achieved by means of diagonal stiffeners, three types of stiffeners are usually considered, K stiffener, N stiffener or Morris stiffener.

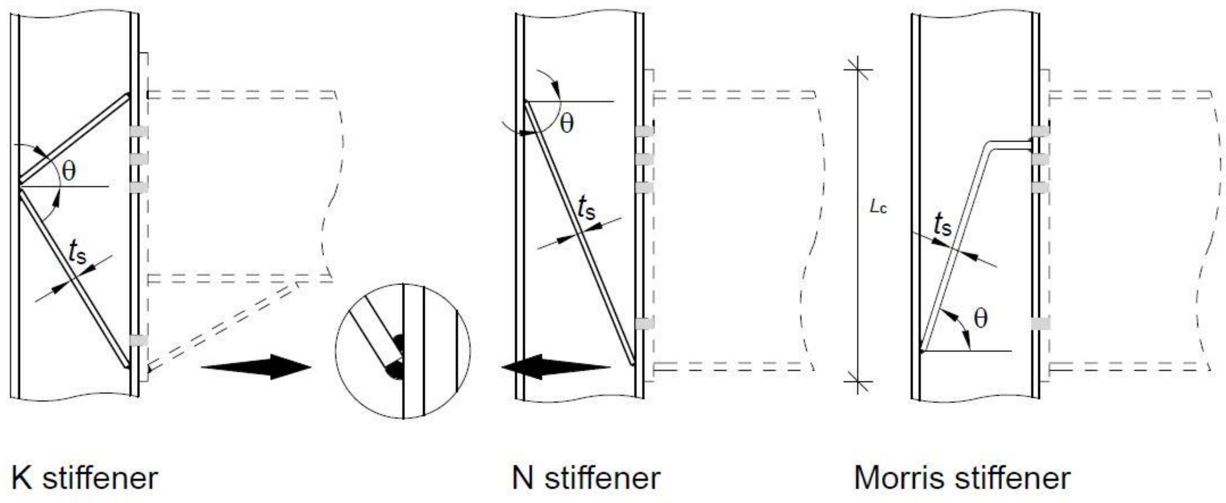


Figure 2.13 – Diagonal stiffeners (SCI/BCSA. (2013))

The K stiffener is usually applied for the cases where large ratios of connection depth to the depth of the column are being analysed.

N stiffeners may be adopted instead of K stiffeners for cases for which the access to tensioned bolts is not ideal.

Both K and N type stiffeners should be designed as compression stiffeners.

The Morris stiffener was developed to diminish the observed difficulties in placing the bolts for the cases where K and N stiffeners are considered. Morris stiffeners are usually combined with compression stiffeners to improve the web resistance in the compression zone.

2.2.7 Welds

The design of a connection must also take into account the design of its welds which should allow for the correct transfer of both shear and tension loads between the elements to be connected. This can be achieved by means of fillet or butt welds.

The shear resistance of a fillet weld can be determined accordingly to expression (2.35)

$$F_{vw,Rd} = \frac{a f_u/\sqrt{3}}{\beta_w \gamma_{M2}} \tag{2.35}$$

Where a is the throat thickness of the weld, f_u is the ultimate tensile strength of the weaker part, β_w is a correlation factor according to the strength of the weaker part and may be taken as 0.85 or 0.90 for S275 and S355 steel grades respectively.

The resistance of a fillet weld to transverse forces is given by,

$$F_{nw,Rd} = K \frac{a f_u / \sqrt{3}}{\beta_w \gamma_{M2}} \tag{2.36}$$

Where $K = \sqrt{\frac{3}{1+2cos^2\theta}}$ and θ is the angle between the transverse force and the throat of the weld as represented in Figure 2.14.

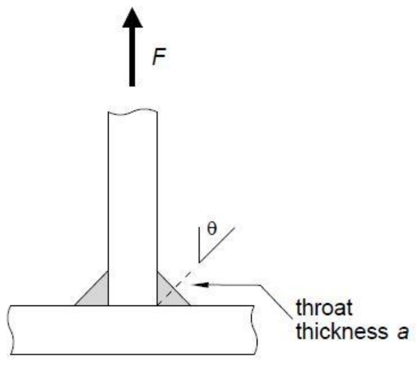


Figure 2.14 – Transverse force applied to a weld (SCI/BCSA. (2013))

It is usual for practitioners to design the welds as a full-strength connection in order to avoid that these component be the conditioning one. When considering fillet-welds, full-strength connections can be achieved by means of a throat thickness of 1.0 or 1.2 times the smallest thickness to be connected for S275 or S355 grade steel respectively. Full penetration butt welds for their turn have a design resistance that may be considered to be equal to the design resistance of the weaker part to be connected.

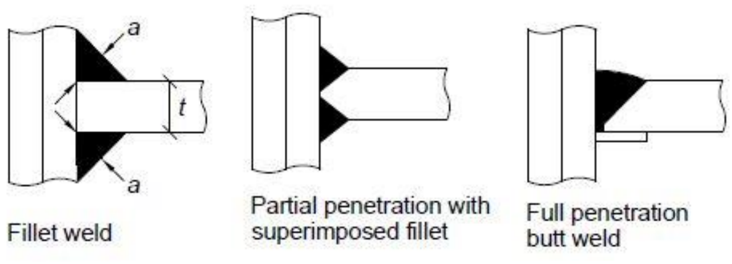

a
t
a
Fillet weld
Partial penetration with superimposed fillet
Full penetration butt weld

Figure 2.15 – Weld types (SCI/BCSA. (2013))

2.3 Summary

Despite rather exhaustive, the design procedure previously presented, is currently applied in a somewhat simple and expeditious manner, being it through the development of self-made spreadsheets, or by means of available software, either one of them comprising a common practice.

However, even a broadly automated task, repeated on a large number of examples, may become a time consuming and prone to error process. In order to bypass this issue, it may be of interest to take advantage of different techniques, not physic based but rather based upon data available from previous examples, such as is the case of the application of learning algorithms, the base for Machine Learning.

3 MACHINE LEARNING

The increasing complexity of a wide range of problems, across numerous fields, may lead to situations for which there isn't a particular method for computing an output based upon the available inputs or situations where the needed computation is much too expensive, making the application of common explicit programming impractical or even impossible. In order to bypass these issues, an alternative approach, in which a computer tries to learn from examples, known as learning methodology, can be taken advantage of.

The various definitions of Artificial Intelligence (AI) laid out by different authors can be divided, accordingly to Russel (Russel, S. J., Norvig, P. (2016)), into four categories (Thinking Humanly; Thinking Rationally; Acting Humanly; Acting Rationally), depending on what concerns each one of them, be it the thought process and reasoning capabilities of a machine, their behaviour or their success in accurately execute different tasks when compared to human performance (Figure 3.1).

Thinking Humanly	**Thinking Rationally**
"The exciting new effort to make computers think…machines with minds, in the full and literal sense." (Haugeland (1985)) "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning…" (Bellman (1978))	"The study of mental faculties through the use of computational models." (Charniak and McDermott (1985)) "The study of the computations that make it possible to perceive, reason, and act." (Winston (1992))
Acting Humanly	**Acting Rationally**
"The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil (1990)) "The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight (1991))	"Computational Intelligence is the study of the design of intelligent agents." (Poole et al (1998)) "AI…is concerned with intelligent behaviour in artifacts." (Nilsson (1998))

Figure 3.1– Artificial Intelligence definitions (adapted from Russel, S. J., Norvig, P. (2016))

Although the first recognized work in AI was done in 1943 by Warren McCulloch and Walter Pitts following the works of Russel, Whitehead and Turing's theory of computation, the field has its official birthplace in Dartmouth, following McCarthy's initiative to organize a two-month workshop in Dartmouth College during the summer of 1956 (Russel, S. J., Norvig, P. (2016)).

Machine Learning (ML) and AI are sometimes sought as being interchangeable concepts. However, the former can be classified as a subfield of the latter (Figure 3.2). The term Machine Learning was coined by Arthur Samuel in 1959 and can be defined as a branch of computer science which goal is to develop models by means of algorithms that can automatically extract patterns from within a dataset and then use the found patterns to make predictions regarding new data (Murphy, K. P. (2012)).

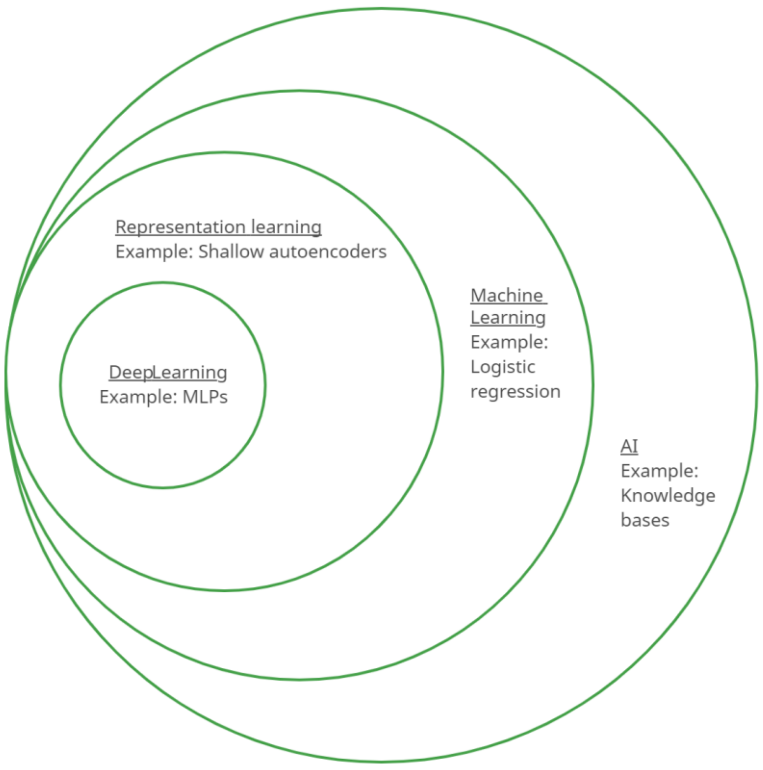


Figure 3.2 – The different levels of AI (adapted from Goodfellow, I., Bengio Y., Courville, A. (2016))

The process involved in the resolution of a certain problem by means of Machine Learning techniques is composed of a set of interlinked steps that result in a workflow such as the one presented on Figure 3.3.
The first step, here named “Get Data”, is one of the most important steps since it is this initial volume of data that will be the basis of all remaining work, and from which will depend on the quality of the created models. Once all the data has been gathered it is usually necessary to preprocess it (“Clean, Prepare and Manipulate Data”) given for instance the existence of incorrect examples, examples with missing information or simply because some algorithms will only work if the data being given as an input is in a specific format.
Once all the dataset related tasks have been completed, it is necessary to choose a model for the respective training process to begin. The original dataset is usually divided in some proportion into a training and a testset. The first is used as an input for the learning algorithm in order to be possible to define the proper parameters enabling the development of a model

with a good quality. Once the training process is completed, it is possible to analyse the quality of the model by comparing its predictions with the actual labels of a held out dataset, such as is the case of the testset. Considering the quality of the results which can be measured by different metrics, it is possible to further tune the algorithm's parameters in order to further improve the quality of the results.

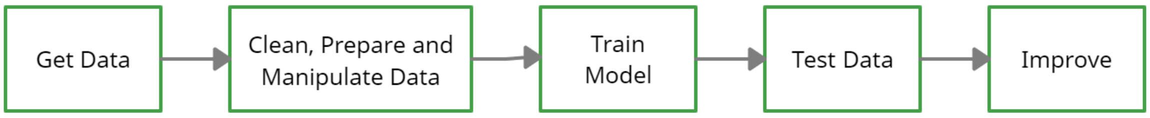


Figure 3.3 – Machine Learning workflow

3.1 Types of learning

Machine learning approaches may be divided into different learning types such as supervised, unsupervised, semi-supervised learning and reinforcement learning.

3.1.1 Supervised learning

Supervised learning is the most used form of ML. In this type of learning, the algorithms are presented with a dataset containing not only the value of the different features $x^{(j)}$ (in the form of a feature vector $\boldsymbol{x_i}$) describing each example but also their respective label or target $y^{(j)}$, thus comprising pairs of input-output (feature-label) which may be given to the machine as a training set in the form of the collection of training examples $\{(\boldsymbol{x_i}, \boldsymbol{y_i})\}_{i=1}^{N}$, where the label vector $\boldsymbol{y_j}$ may comprise, for example, a finite set of classes $\{1,2,\dots,C\}$, real numbers or even other more complex structures.
Supervised learning, as the term itself suggests, consists in showing the machine what to do by providing the "answer" $\boldsymbol{y_j}$, as a teacher would do, or in this case, a knowledgeable external supervisor. The objective of the machine is thus to find a hypothesis h (or a function $f(x)$) from a training set that should enable a close agreement with the known labels in order to allow for posterior predictions on new inputs, a process known as generalization (Russel, S. J., Norvig, P. (2016)).
One of the best-known and oldest examples of supervised learning task is the application to the Iris dataset, which comprises a collection of 150 examples of Iris flowers, namely the measurements of some of their characteristics (features) such as the sepal length, sepal width, petal length and petal width, together with the respective species classification.

3.1.2 Unsupervised learning

As opposed to supervised, in unsupervised learning, the algorithms are presented with datasets comprised of unlabelled examples $\{(\boldsymbol{x_j})\}_{i=1}^{N}$ thus being impossible to provide the algorithm with any kind of "answer" $\boldsymbol{y_j}$ or desired output. The objective of this kind of learning is to find properties of interest and meaningful information such as patterns within the structure of our data in order to create a model that takes an input such as a feature vector and transforms it into an output that may be used to solve a practical problem.
One of the most common examples of unsupervised learning is the task of clustering. This kind of tasks consist in organizing a large quantity of information into subgroups known as clusters that comprise a group of elements that are similar between themselves but whose characteristics are different from other clusters.

3.1.3 Semi-supervised learning

As the term suggests and following the previous definitions semi-supervised learning falls somewhere in between unsupervised and supervised learning. In semi-supervised learning, the provided dataset is comprised of both labelled and unlabelled examples, many of the times in an unbalanced way, with the unlabelled examples largely outnumbering the labelled ones. Since the 1990's, there's been a growing interest in semi-supervised learning, especially applied to natural language and text classification problems. Although it may seem counterintuitive, there are cases for which the application of semi-supervised learning yields better results than if supervised learning is applied, in particular, those for which "the distribution of examples, which the unlabelled data will help elucidate, be relevant for the classification problem". (Chapelle, O., Schölkopf, B., Zien, A. (2006)

3.1.4 Reinforcement learning

In reinforcement learning a system often known as a decision-making agent is able to improve its performance by maximizing the expected reward in order to achieve a certain goal. These different rewards are a result of the actions executed by the machine in its interaction with its surrounding environment.
Usual examples of reinforcement learning comprise chess engines and other kinds of game playing, where the reinforcement is the winning or losing of the game, as well as robotics or logistics, where the reinforcement can be accomplished or not of a given task (Sutton, R. S., Barto, A. G. (2018)).

3.2 Types of Problems

3.2.1 Classification

In a classification task, the goal of the learning algorithm is to find the class, among a finite set of C discrete classes, to which an unlabelled example belongs to, based on past observations. In order to do so, the algorithm looks for a function $f: \mathbb{R}^n \longrightarrow \{1, \dots, C\}$ which assigns a feature vector $\boldsymbol{x}$ to the class y, $y = f(\boldsymbol{x})$. If a single label vector $\boldsymbol{y_1}$ comprises just $C = 2$ classes, then the problem at hand may be defined as a binary classification problem, otherwise, for the cases in which $C > 2$, the classification is defined as being multiclass. Furthermore, if more than a label exists for each example and their classes are not mutually exclusive, the problem is referred to as a multi-labelled one.
A typical example, referred in Section 3.1.1, is the classification of the Iris flowers, where the goal is to determine the type of Iris (Setosa, Versicolour, Virginica) each example is.

3.2.2 Regression problems

The objective of regression problems is to predict a target in the form of a numerical continuous value, by giving the model an input. This task is similar to classification problems, although with a different output format and can be achieved by looking for a function $f: \mathbb{R}^n \longrightarrow \mathbb{R}$.
Regression tasks are commonly used by insurance companies in order to predict the amount of claims that might be made by an insured person (Goodfellow, I., Bengio Y., Courville, A. (2016)).

3.2.3 Clustering

Clustering learning problems, which can usually be solved by means of algorithms such as the k-means clustering algorithm, consist upon the division of a dataset into clusters composed of similar examples. Contrary to regression and classification tasks, in clustering problems, the training examples do not have an associated label, being the algorithm's objective, the creation of clusters and thus an implicit definition of classes through the definition of the corresponding groups of examples (Silva, C., Ribeiro, B. (2018)). Hence, clustering is not only inserted in the set of unsupervised learning but is also unsupervised learning's most common type of problem.
A usual example of the application of this type of learning task is the creation of groups of users of e-commerce platforms, based upon their purchasing behaviour, which will enable these platforms a targeted advertising strategy (Murphy, K. P. (2012)).

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

3.2.4 Anomaly detection

Anomaly detection tasks, also known as outlier detection, consists in the analysis of a problem by means of an algorithm that identifies examples that correspond to abnormal or atypical cases and that are thus identified for being considerably different from the remaining examples composing the dataset.
This type of task is not usually defined as a supervised classification learning problem, composed of two classes, one corresponding to the so-called normal examples and the other to the outliers, since usually, the number of examples that could be classified as outliers is very low and thus it cannot compose a pattern that may be easily found by a two-class classifier.
Anomaly detection techniques are usually deployed by companies in tasks such as credit card fraud detection (Goodfellow, I., Bengio Y., Courville, A. (2016)).

3.3 Data Preprocessing

Data preprocessing is a process in which a feature or group of features is extracted or modified from a raw dataset in order to obtain a final dataset more suitable for a particular machine learning model, thus enabling higher quality outputs (Zheng, A., Casari, A. (2018)).

3.3.1 Normalization

For the case of the more common data type, numerical data and depending on the chosen machine learning algorithm, it may be necessary to consider among others, the scale of the inputs, which may lead to the need to normalize the features, a process where the original range of values of a feature is converted into a standard range such as [0;1] (Burkov, A. (2019)).

3.3.2 Standardization

The statistical distribution of features in a dataset, or the probability that a feature takes a particular value, might also be of importance for the performance of the ML algorithm. In these particular cases, a standardization procedure should be considered. The standardization of a feature consists in rescaling it in order to give it the properties of a standard normal or Gaussian distribution with $\mu = 0$ mean and $\sigma = 1$ standard deviation.

3.3.3 Binning

It may also be of interest in some cases to convert a numerical feature into a categorical one in which case a binning or bucketing process, consisting of converting a continuous feature into different value ranges is applied. This process can be better understood considering a task where a feature comprising the age of a group of people represented by real value numbers, is divided into different bins with 0 to 10 years-old on a bin, 10 to 20 years-old on the following bin and so on.

3.3.4 One-hot encoding

As opposed to the binning process mentioned above, when a particular feature is comprised of different classes, thus corresponding to a so-called categorical feature, and the intended learning algorithm only accepts as input numerical feature vectors, the later should be transformed into different binary features, leading to an increased dimension of the feature vector, in a process known as one-hot encoding (Burkov, A. (2019)).

In some cases, due to the data gathering process or during the construction of the dataset, especially when this dataset is obtained through a handcraft approach, the values of some features may be omitted. In order to tackle this issue, different approaches are usually followed, among them, the complete dismissal of the examples for which one or more features are missing and the application of data imputation techniques, the later consisting of replacing the missing values by either the average value of the feature, a value outside the normal range of the feature or still a value in the middle of the normal range allowing that the missing value won't significantly affect the prediction.

Data preprocessing may also be of interest for analysis in which the dataset comprises a considerable number of repeated examples and which for that reason may be removed.
The presence of irrelevant features or features with high correlation among each other may allow a reduction of the overall size of the dataset without an impact in the output's quality (Silva, C., Ribeiro, B. (2018).

3.4 Training Models

The no free lunch theorem, a well-known theorem within the field of machine learning (Wolpert, D.H.(1997)), states that all algorithms have the same performance when one considers all possible problems and available data. However, for specific problems, a selection process should be pursued in order to find the most suited algorithm and their respective settings.
Unless one has the time to apply them all, the selection of machine learning algorithms for the task in hands is not an easy process. This selection may be guided by a process of

performance assessment, in which a quantitative measure of the quality of a model can be obtained and be used for an informed decision (Burkov, A. (2019)).
Different models have widespread levels of complexity and the decision to pick one of them can depend upon different factors. If the data is highly nonlinear, the choice for an algorithm which is only able to handle linearly separable data can lead to a poor performance. The explainability of an algorithm may be important for specific tasks and divide the different algorithms into "white-box" and "black-box" algorithms. With "white-box", the users can have a clear view of the decision-making process followed to obtain the wanted outputs which also allows their validation. "Black-box" algorithms such as neural networks, although they may be very accurate, the obtained outputs can be hard to explain and understand.
As mentioned before, the available dataset can be comprised of different data types (numerical, categorical) which may not be accepted by different algorithms, or which might require a pre-processing in order to be accepted.
The training time and prediction speed can also be important factors to consider while choosing the proper machine learning algorithm.

3.4.1 Hyperparameters

Hyperparameters can be understood as tunable settings of the different learning algorithms which are neither directly estimated nor optimized by the machine learning algorithm itself, although it is possible to develop procedures in which the hyperparameters of the former learning algorithm are tuned and optimized by another algorithm (Goodfellow, I., Bengio Y., Courville, A. (2016)). Usually, the process of tuning the hyperparameters is done by searching the combination of values that yields the best result through a grid search procedure. For the cases for which the number of hyperparameters make the grid search procedure impractical, different techniques may be adopted such as a random search, genetic algorithms, among others (Burkov, A. (2019)).

3.4.2 Overfitting and Underfitting

The main objective of the learning algorithm in an early phase of the learning process consists of properly fitting the model to the training data. If the obtained training error, computed on the training set yields unacceptable results, meaning that the built model fails in predicting the outputs of the set in which it was trained, the model is said to be underfitting, which may be a result of an insufficiently complex model or uninformative features.
Once the issue of underfitting has been overcome, the goal of the model becomes to correctly predict new examples. When the complexity of the chosen model is such that not only it learns the patterns present in the dataset but also the particular characteristics of each

example, it may behave poorly in the presence of new data in which case it is said to be overfitting.

3.4.3 Training, Validation and Test Sets

In order to choose the appropriate machine learning algorithm for the specific task being undertaken, it is usual to split the available dataset into different subsets.

The first subset, known as the training set, is used to fit or build the model and is usually comprised of 70% to 80% of randomly obtained examples of the original dataset, although other split ratios can be considered.

The remaining 20% to 30% of the data is usually equitably divided between the validation and test sets. The former is used to estimate prediction error for model selection and tuning the appropriate values of hyperparameters and the latter to assess the generalization error of the chosen model.

Alpaydin (Alpaydin, E. (2014)) uses a clear, easy to understand analogy in order to explain the difference between the different sets: "…when we are taking a course: the example problems that the instructor solves in class while teaching a subject form the training set; exam questions are the validation set; and the problems we solve in our later, professional life are the test set".

3.5 Learning Methods

3.5.1 Linear Regression

Linear regression models are one of the most commonly used forms of supervised machine learning algorithms owing to their simplicity and interpretability, while also being very effective in many different tasks.

These models are the basis for all the other regression methods and have their simplest expression in the form of the simple linear regression which can be written in the form

$$y(x) = \beta_0 + \beta_1 x + \epsilon \tag{3.1}$$

where $E(Y|X = x) = \beta_0 + \beta_1 x$ is the mean function, and ϵ is the statistical error to take into account the difference between the actual response true and the predicted results. β_0 and β_1 are usually unknown parameters that characterize a model, and which may be estimated from the data through different methods. These parameters are known, respectively, as the intercept and the slope and are represented in Figure 3.4.

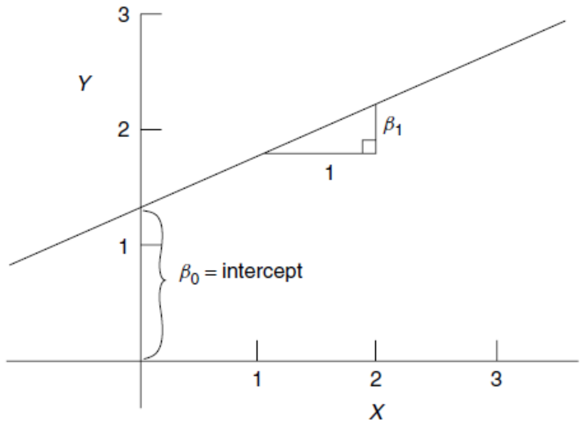


Figure 3.4 – Simple linear regression visualization (Weisberg, S. (2013))

The most usual method used for the estimation of the intercept β_0 and slope β_1 parameters is the ordinary least squares method (OLS).

For the more common cases in which a model contains a set of different variables, a generalization of the simple linear regression, known as multiple linear regression, must be developed, and thus

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{\epsilon} \tag{3.2}$$

where $\boldsymbol{Y}$ a $N * 1$ vector, represents the resultant, $\boldsymbol{X}$ is a $N(D+1)$ matrix of features, $\boldsymbol{\beta}$ is a $(D+1) * 1$ vector of regression coefficients or weights and ϵ is the residual error between the predictions and the respective response as mentioned for the simple linear regression case. It is also usual to find the previous expression written in the form

As mentioned above, the estimation of the coefficients β can be made, among others, by means of the least squares method also known as maximum likelihood estimation. The goal of the least squares linear regression is to minimize the sum of squared errors (SSE) sometimes also called residual sum of squares (RSS), which corresponds to minimize the sum of the lengths of the vertical (black) lines between each training value (red dots) and the respective prediction as represented in Figure 3.5.

$$SSE = \sum_{i=1}^{N} (y_i - y(x_i))^2 = \sum_{i=1}^{N} (y_i - \sum_{j=1}^{D} w_j x_{ij} - \epsilon)^2 \tag{3.3}$$

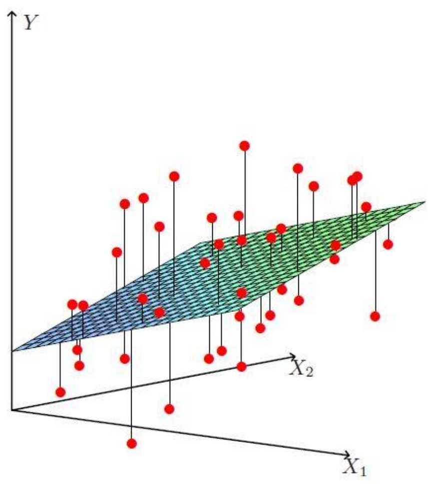


Figure 3.5 – Linear least squares fitting (Hastie, T., Tibshirani, R., Friedman, J. (2013))

3.5.2 Logistic Regression

Although this type of model has the same objective as that of all other regression models, and despite being called a regression, this method is in fact a form of classification (Murphy, K. P. (2012)) and not a regression model since the obtained responses or outputs correspond to discrete binary values which can be understood as categories such as true or false and success or fail.

The bounded nature of the outputs involved in Logistic Regression problems, means that the application of the ordinary least squares method (OLS) to the data is not the correct approach as this would allow the return of values outside the expected range.

In order to allow for a continuous function with a codomain [0,1] that represented the above-mentioned categories such as true or false if an input is close to 1 or 0 respectively, and considering its mathematical flexibility and the interpretability of its parameters, the sigmoid or standard logistic function was adopted,

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.4}$$

where x is the input and e is Euler's number which constitutes the base of the natural logarithm.

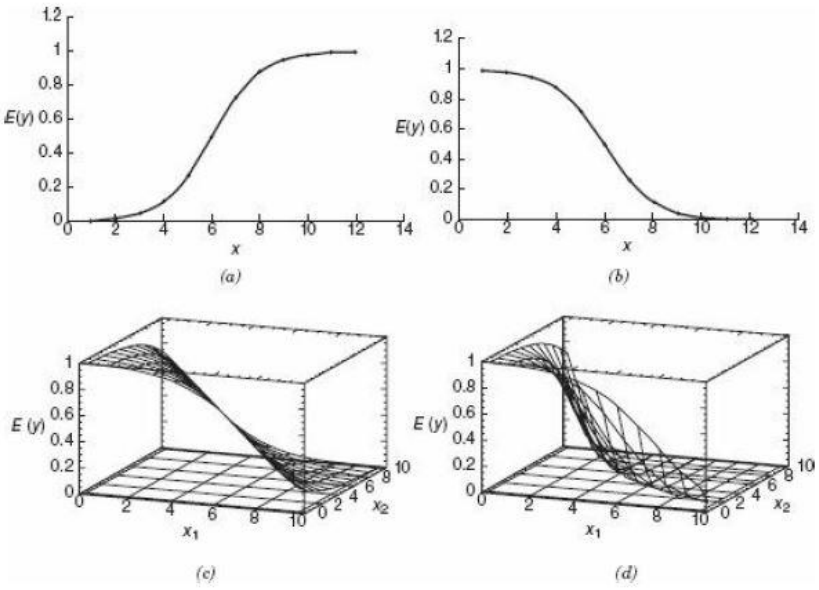


Figure 3.6 – Logistic functions (Montgomery, D. C., Peck, E. A., Vining, G. G. (2015))

The Logistic Regression model to be fitted to the data has the form,

$$sigm(\beta_0 + \beta_1 x) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}} = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}} \tag{3.5}$$

where β_0 and β_1 are the unknown parameters which for Linear Regression models were estimated by means of the least squares method by minimizing the sum of squared errors observed between the predicted values and the actual response. In Logistic Regression models, the estimate of the unknown coefficients is made by means of the maximum likelihood method, which maximizes the probability of returning the actual data and demands the construction of the likelihood function which represent the probability of the observed data as a function of the unknown parameters.

3.5.3 Decision Trees

This type of learning technique addresses an originally complex problem by splitting it into a variety of smaller, simpler problems, in a hierarchical structured divide-and-conquer-like approach, constituting a very simple and interpretable algorithm which can be summed up into a set of if-then rules (Silva, C., Ribeiro, B. (2018)) and can be applied to both classification and regression problems.

A decision tree can be defined as a nonparametric method which divides the feature space into local regions to which a class is associated. Decision trees are composed of decision nodes m which comprise a test function $f_m(\boldsymbol{x})$ from where different branches emerge, each one of these branches corresponding to a possible feature value or range of values (Alpaydin, E. (2014)).

The first step is to select a feature to be placed at the first decision node, usually known as the root node, and to create a branch for different values of the feature thus splitting the problem into subsets, a process which can be recursively repeated until the leaf nodes, at the base of the decision tree are achieved, and thus an output is obtained. Each path followed between the root node and the different leaf nodes corresponds to a classification rule.

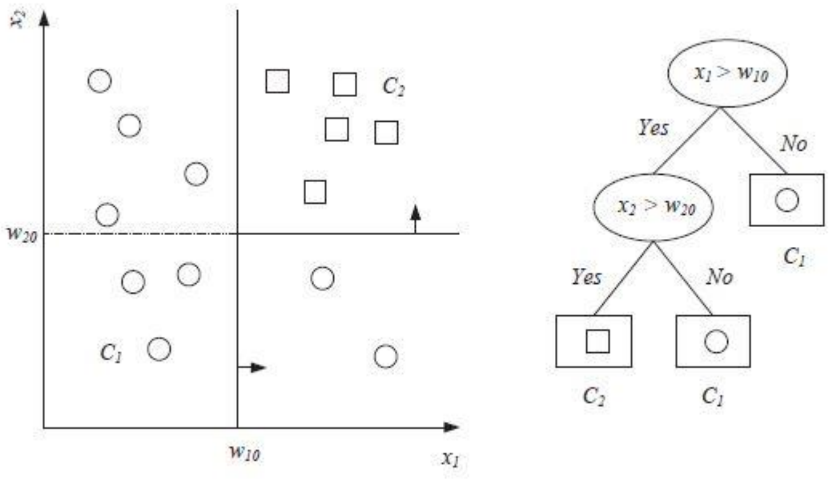


Figure 3.7 – Dataset partition and corresponding decision tree (Alpaydin, E. (2014))

The goal while constructing a decision tree is to obtain the shortest possible decision tree, and thus to arrive to the leaf nodes in the shortest number of decision nodes, which can be achieved by testing the most important attribute first and hence by looking at information I or entropy, a measure of the randomness of a variable to be predicted, in this case, a class. If for a particular learning problem exists an optimum attribute, it would divide the examples into subsets that are all positive or all negative while if an attribute generates subsets with a similar proportion of negative and positive examples as that of the original set, it would be a rather useless one.

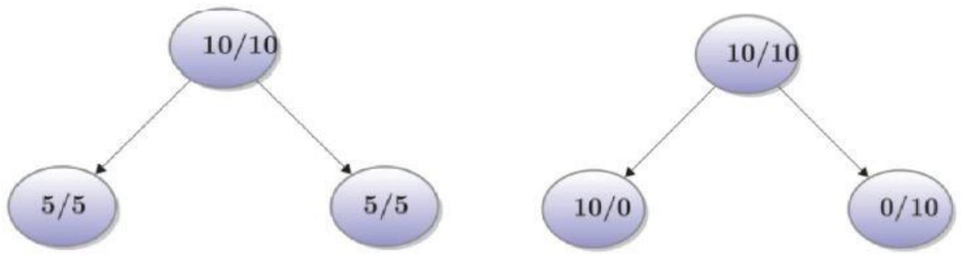


Figure 3.8 – Choice of attribute and obtained subset distribution (Silva, C., Ribeiro, B. (2018))

Information I is measured in bits and can be computed by counting the number of yes/positive p or no/negative n classes ate a node and dividing each one by the total number of possibilities, leading to their respective probabilities $P(v_i)$.

$$I(P(v_1), P(v_2), \dots, P(v_n)) = \ entropy(P(v_1), P(v_2), \dots, P(v_n)) = -\sum_i^n P(v_i) log_2(P(v_i)) \quad (3.6)$$

where v_n is the nth possible answer of the random variable

Information I is usually also referred to as entropy. It reaches its maximum of $log_2(P(v_i)$ when the possible values of the random variable of interest I are equiprobable $P(v_i) = P(v_j)$ for any i different from j, and its minimum of $I(X) = 0$ if there is any i for which $P(v_i) = 1$, corresponding to the cases for which the random variable only have a single value and thus all examples have the same label.

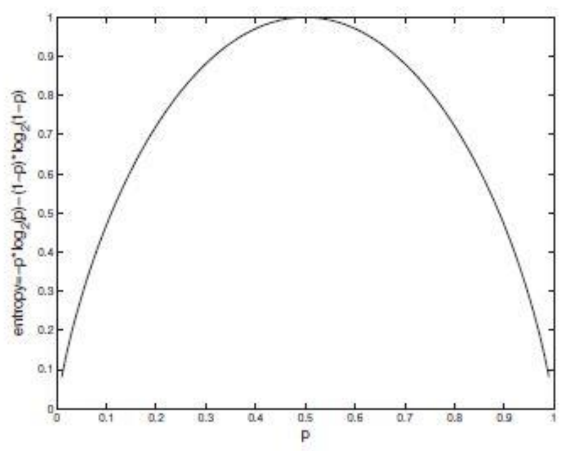


Figure 3.9 – Entropy function (Alpaydin, E. (2014))

The reduction of entropy due to the partitioning of the examples in its turn can be measured by the information gain achieved from the feature or attribute A in each decision node.

$$Gain(A) = I(P(v_1), P(v_2), \dots, P(v_n)) - \sum_{i=1}^{v} \frac{v_i}{v_n} I(P(v_1), P(v_2), \dots, P(v_v)) \quad (3.7)$$

where v_v is the vth possible answer of a subset obtained from the division of the feature considered on a decision node to the ones below.

There are some criteria that can be considered in order to define the moment for which the division process should be stopped. The first one corresponds to the case for which all examples belong to a single class and thus the leaf node is pure.
In some other situations a certain minimum number of instances entering a node may be required, since a generalization error may be incurred due to decisions which are based upon

an insufficient number of examples. This process known as prepruning simplifies the tree. A minimum value for the reduction of entropy during the splitting process of a decision node may also be considered as a stop criterion for the construction of a decision tree.

3.5.4 Random Forests

This algorithm falls in a particular learning method known as ensemble learning in which a set of different models learned from the data are used in order to achieve the intended goal (Silva, C., Ribeiro, B. (2018)). This approach is usually compared to typical human decision-making processes in which the opinion of multiple experts is taken into account and eventually combined in order to achieve a consensus or, in the cases for which this is not possible, a voting procedure is engaged upon.

The ensemble learning method is thus based upon the assumption that a set of k learning algorithms will achieve better results than a single learning technique, if the obtained outputs are properly combined, being the combination of the different algorithms achieved by means of voting algorithms.

There are several reasons that may ultimately guide the practitioner to the use of ensemble learning such as reducing the risk of having a bad performance in the test set by combining the outputs of an array of different algorithms which make different assumptions about the presented data, or by using an array of the same learning technique although with the use of different values for the corresponding hyperparameters.
In particularly large datasets, while the use of a single model may be inefficient, the partition of the dataset into random subsets and the use of different individual classifying algorithms in each subset, in a process called bagging, may make the global analysis a more efficient one, reducing variance and overfitting. In bagging also known as bootstrap aggregation, the above-mentioned subsets are obtained from the training set with replacement of the chosen examples, enabling that different subsets may contain the same example. The output class is determined from the individual classifying algorithms by choosing the one that was obtained from the latter the most number of times.
It may also be of interest to apply different learning techniques in a serially manner, which can be made among others through boosting or cascading, allowing for the current running algorithm to pick up and give emphasis to instances that previous models failed to learn.
As mentioned for the case of decision trees, one way to solve complex problems is to follow a divide-and-conquer approach, this also applies for ensemble learning, in particular in situations where a set of different algorithms are capable of solving issues that a single algorithm cannot solve.

Random forests can be applied to both classification and regression problems and are based upon the bagging technique mentioned above, consisting in a process where multiple decision trees are built and for which the output is the value that appears most often, known as mode, for classification tasks, and the mean of the predicted value for each decision tree in the case of regression problems.
The different subsets used by this learning technique should comprise about 66% of the total number of training examples each. The number of examples to be considered in a subset and the number of decision trees constitute the most important random forest's hyperparameters.
For each training subset, a subset of m features are randomly chosen, being the one with the most information gain placed at the root node, as usually done for the case of individual decision trees. The random selection process of features avoids possible correlations between decision trees, increasing the overall accuracy of random trees.

3.5.5 k-Nearest Neighbors

The k-Nearest Neighbors algorithm, also known as kNN is another simple and explainable learning technique, one that is instance-based meaning that unlike other type of machine learning algorithms where a model is fitted to the dataset allowing the training examples to be discarded after, in this technique the training instances have to be stored since they are only used when a new example has to be classified, and thus a model is never really built. This leads to a process for which the computational power is deployed mainly in the test stage while the training process is almost non-existent. This type of processes is sometimes slow and with a duration that is proportional to the number of training examples, demanding larger computational efforts being usually inserted in the lazy learning framework.

The kNN learning technique uses the k nearest examples of the training set to classify a new unseen input, based upon a specific distance, considering for this effect the predominant class in the set of the k nearest training elements.
Considering the previous description of the procedure used by the kNN approach, there are a few key considerations to be made, namely the use of representative samples instead of the full training set depending on the number of examples, the choice of the number of the k nearest examples, the type of distance used to measure the proximity of the training points to the new unseen data and the method used to classify the new input.

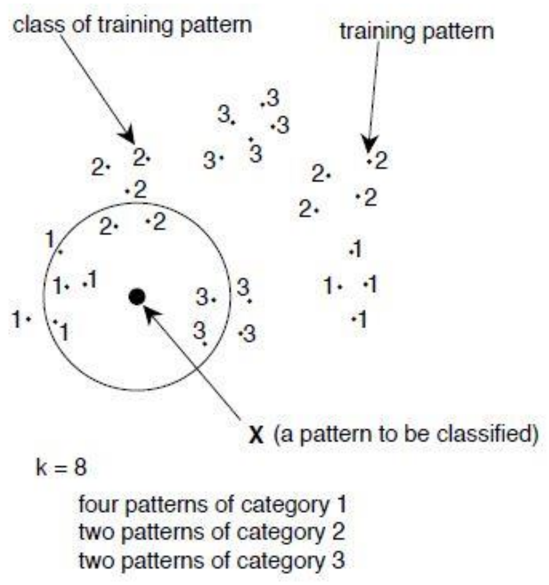

class of training pattern
training pattern
X (a pattern to be classified)
k = 8
four patterns of category 1
two patterns of category 2
two patterns of category 3

Figure 3.10 – kNN example (Nilsson, N. J., (2005))

The definition of the value k is no easy task demanding, many times, the use of a tuning procedure to find the best value depending on the problem in hands. Taking into account the example defined by Figure 3.11, where three different options are considered for the value of the k nearest neighbors. For $k = 1$, thus a small value, the output would be a square and it may be considerably influenced by noisy data. For a larger $k = 7$ the obtained answer would be the class of triangles, while increasing even further the number of $k = 15$ the output would once again be the class of squares, being obvious the influence of data points of different classes when considering large values of k. The values considered for the number of neighbors k should always correspond to an odd number in order to avoid ties, in which case the obtained output would be a random value between the tied ones (Kuhn, M., Johnson, K. (2018)).

Different advantages and disadvantages can be pointed out when using small or large values of k. When considering the latter, as expected, the computational effort demanded by the algorithm is greater, enabling however smoother decision boundaries and the extraction of probabilistic information due to the availability of the proportions of examples that belong to the different existing classes.

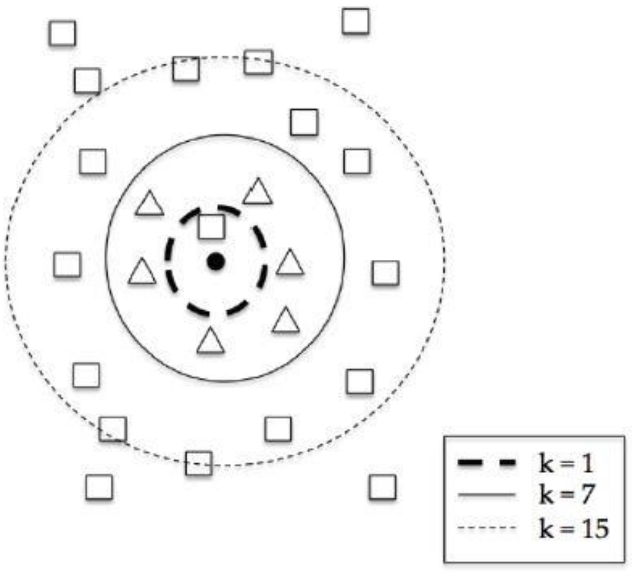


Figure 3.11 – Influence of the value k (Silva, C., Ribeiro, B. (2018))

The distances used to evaluate the nearest neighbors can be computed in different ways, being the Euclidean and the Manhattan the two most common used distances.
The well-known Euclidean distance corresponds to the shortest distance between two points x and y and may be defined as,

$$d(x,y) = \sqrt{\sum_{j=1}^{n}(x_j - y_j)^2} \tag{3.8}$$

The Manhattan distance between the two abovementioned points, can be interpreted as a path with 90° turns, which can be computed as,

$$d(x,y) = \sum_{j=1}^{n}|x_j - y_j| \tag{3.9}$$

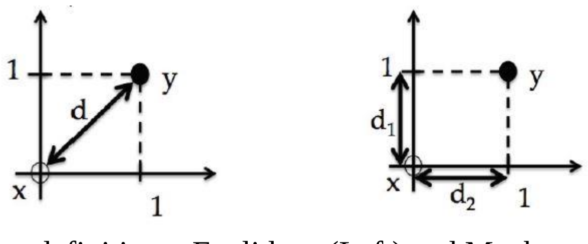


Figure 3.12 – Distance definition – Euclidean (Left) and Manhattan (Right) (Silva, C., Ribeiro, B. (2018))
The classification of an example based upon the class of the k nearest neighbors, can be done through different approaches, being the simplest one known as the majority voting, which accordingly to its name consists in assigning to the new example the class which is the most represented by the examples that constitute the k nearest neighbors. Another more complex approach is to assigned different weights for the different neighbors, proportional to their

distance to the new example. Since the closest neighbors should represent better the class to which the new example belongs, their contribution, and hence their weight, for the classification of the example should be greater and thus one possible weight to be considered could be $\frac{1}{d(x,y)^2}$ (Silva, C., Ribeiro, B. (2018)). The latter procedure also contributes in making the obtained output less vulnerable to fluctuations of the k value.

3.5.6 Neural Networks

Artificial neural networks (ANN) were developed following a growing interest regarding the functioning of the human brain and the realization that this process could be efficiently replicated by computers.
The human brain is comprised of a set of 10^{11} individual simple elements known as neurons, which are interconnected in such a way that each neuron is linked to 10^4 of its peers, forming a web that allows the execution of complex tasks and decisions in an efficient and rapid way. This performance is a result of a neuron's low switching times to excitations received from the connections to other neurons, a process that occurs in 10^{-3} seconds, and that when compared to the switching speeds of computers 10^{-10} seconds seems rather slow.
Taking into account a neuron's switching time and that for example a 10^{-1} seconds time span is needed to recognized our loved ones, it would be expected that only a few hundreds of the 10^{11} neurons would be involved in this processes, leading to the assumption by the scientists that the brain's functioning is a result of a synergy obtained from a large quantity of neurons working in parallel instead of the sum of each neuron's individual action.
Nowadays scientists believe that a part of the biological neural network is born with each subject, while the rest is a result of the different experiences and interactions to which one is subjected to, leading to the creation of new connection resulting in new learnings, memories and other basic neural biological functions.
Figure 3.18 represents a single neuron and its structure which can be divided up into a Dendrite, the body cell or Soma and Axon. The first act as receivers and are comprised of nervous cells that carry electrical signals to the Soma where the sum of the different received signals is carried out. Depending on the value of this sum, if a certain threshold is achieved, the signal is sent to the Axon, otherwise the neuron won't transmit any information down the network and thus won't excite the neurons to which it is connected. When the defined threshold is met, the signal is passed to other neurons through the synapses, being the connection with different neurons as strong as the respective synapse which ultimately depends upon the value of the sum received from the Soma.
As ANNs are based upon biological neural networks, they exhibit a considerable number of characteristics that can be observed in human cognitive processes, namely, the ability to learn from experience, the possibility of generalization from examples and the capacity of abstraction regarding characteristics that only contain irrelevant facts for the situation in

hands. Furthermore, this type of learning technique is fault tolerant, leading to a considerable adaptation capacity.

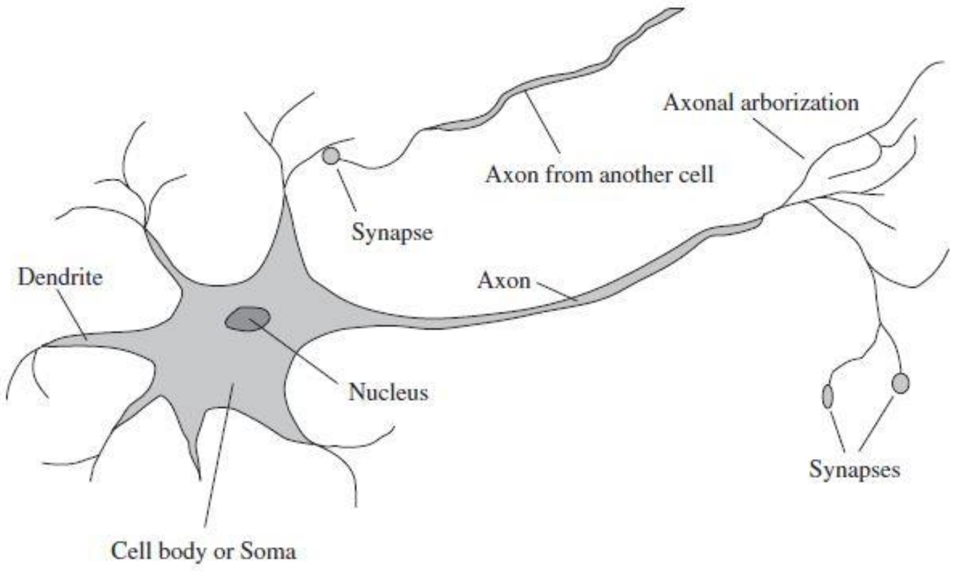


Figure 3.13 – Neuron (Russel, S. J., Norvig, P. (2016))

Artificial neural networks (ANN), a field which is sometimes also mentioned as parallel distributed processing, connectionism or neural computation, can be understood as a network analogous to the biological neural networks described previously, in which the neurons are represented by basic units known as perceptrons.

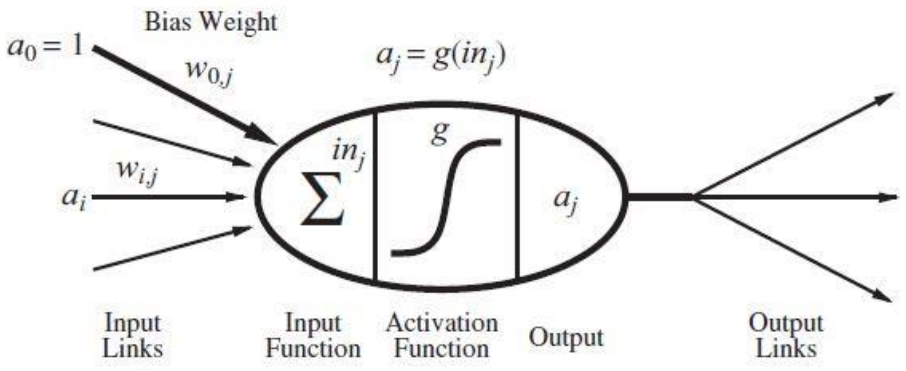


Figure 3.14 – Perceptron (Russel, S. J., Norvig, P. (2016))

A perceptron j has as its goal the calculation of a linear combination by means of a weighted sum in_i of the activation values a_i (Equation 3.10) in order to obtain an output which can take the values 0 or -1 depending on the activation value determined from an activation function $g(in_i)$.

$$in_i = \sum_{i=0}^{n} w_{i,j} a_i \tag{3.10}$$

where a_i is the so-called activation received from unit or perceptron i and $w_{i,j}$ is the weight associated to link from perceptron i to j, and that determines both the sign and strength of the connection.
In addition to the different weights associated to the different links it is also taken into account an additional weight $w_{0,j}$ known as bias weight which defines perceptron j's threshold.

The goal of the activation function, as the name suggests is determine if the weighted sum in_i activates or inactivates the perceptron. Furthermore, it is usual to define the activation function as a nonlinear function due to the fact that if a linear function is considered, the entirety of the neural network is resumed to a linear function.
Considering the threshold function as the activation function, the activation process could be represented as,

$$\begin{cases} 1 \; for \; \sum_{i=0}^{n} w_{i,j} a_i \geq 0 \\ 0 \; for \; \sum_{i=0}^{n} w_{i,j} a_i < 0 \end{cases} \tag{3.11}$$

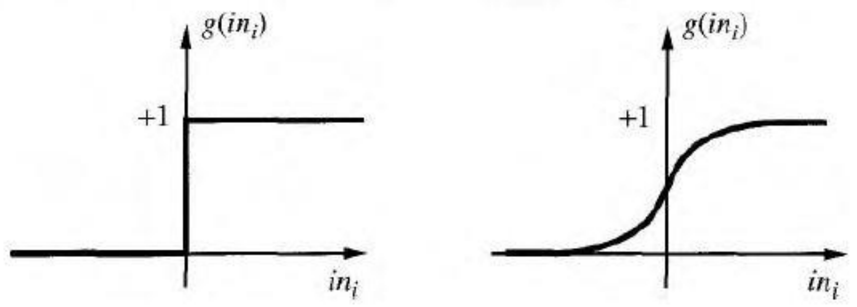


Figure 3.15 – Activation functions. Threshold function (left), Linear function (middle), Sigmoid function (right) (Russel, S. J., Norvig, P. (2016))

Despite being able to represent simple logic operations such as the And and Or Boolean functions, a single perceptron is limited to the representation of linearly separable functions, which is a serious limitation, one that demands the construction of a network of perceptrons with different structures.

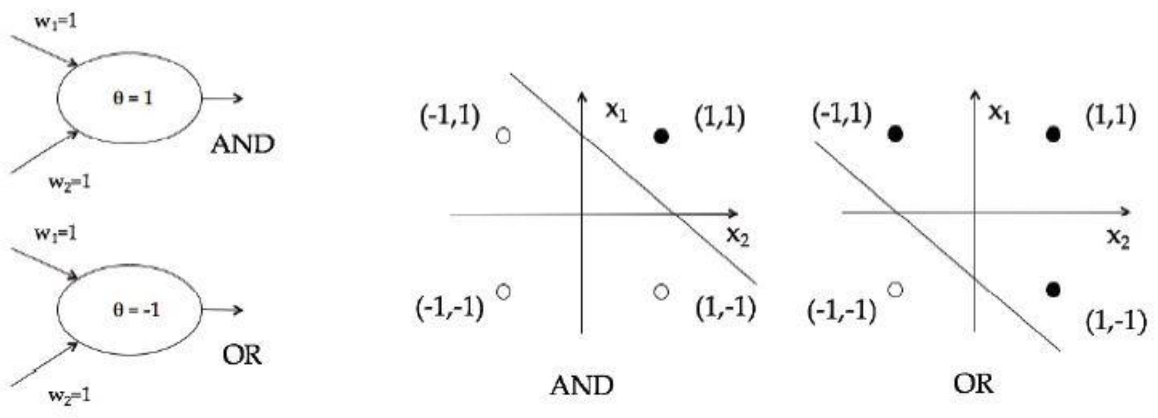


Figure 3.16 – And and Or Boolean fuctions (Silva, C., Ribeiro, B. (2018))

Accordingly to whether the data can follow only a direction along the network or there will be a loop, the ANN can be classified as a feed-forward network or as a recurrent network.
Feed-forward networks, as the name implies, are networks that are comprised of connections that only go along one way, receiving inputs upstream and delivering outputs downstream, thus representing itself a function of the inputs.
Recurrent networks on their turn, allow as inputs their own obtained outputs, resulting in a more complex system for which its response depends upon not only on the fed inputs but also upon the initial state of the inputs obtained from the outputs making them harder to understand.

The example presented in the figure below represents a 3-layer artificial neural network. The first layer, known as the input layer, is comprised of two inputs x_1 and x_2, each connected to a middle layer, in this particular case a single hidden layer composed of two perceptrons n_1 and n_2. The output layer is comprised, in this example, of a single perceptron n_3 with a single output.

While the definition of the number of input and output perceptrons only depends upon the number of available features and intended outputs, the number of hidden layers is of harder definition, being able to influence not only the algorithms capacity to solve the problem if an insufficient number of layers are chosen but also its generalization capabilities if too many hidden layers are considered.

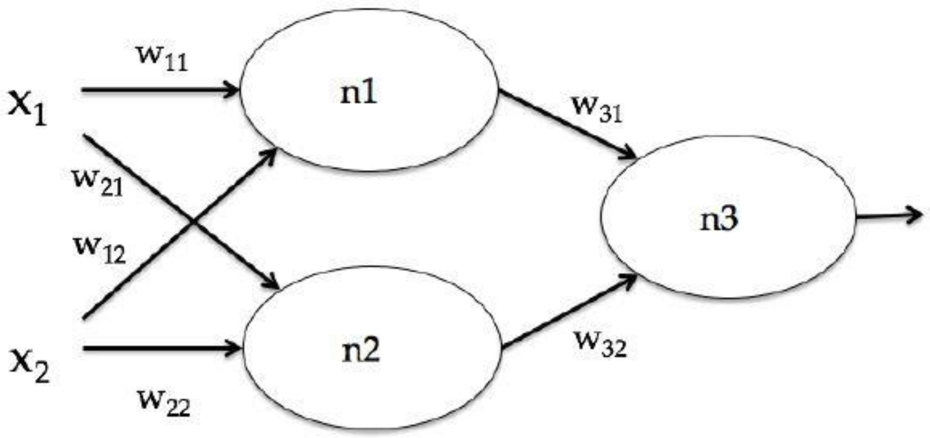


Figure 3.17 – Neural network with 3 layers (Silva, C., Ribeiro, B. (2018))

In order to simplify the understanding of the procedure of finding the weights to be considered in a neural network, a simple single layered network, usually known as single-layer feed-forward neural network or perceptron network is considered.

The goal of the training is to determine the weights that lead to the determination of correct output for the different training examples. Two possibilities are usual considered, the perceptron training rule and the delta rule.

The perceptron training rule begins with a random set of values for the different weights. This are then altered every time a example is misclassified, a process repeated until the perceptron is able of classifying every training example, being the different weights updated accordingly to,

$$w_{i+1} = w_i + \Delta w_i \tag{3.12}$$

$$\Delta w_i = \eta(t - o)x_i \tag{3.13}$$

where t is the target output, o is the output generated by the perceptron and η is a positive constant known as learning rate, whose function is to moderate the rate at which the weights may vary at each training example. Considering the definition of Δw_i it is easy to conclude that if an example is correctly classified then $t = o$ and $t - o = 0$ and thus $\Delta w_i = 0$, leading to no updates on the value of the respective weight.

Despite of the intuitive nature of the perceptron training rule, and its ability to converge in a finite number of steps, for a small enough value of η, this will only happen if the displayed examples are linearly separable, otherwise, convergence is non-guaranteed.
To solve this issue the gradient descent-based delta rule was developed, allowing for the determination of the weight vector that best suits the intended outputs.

The process starts with the definition of a measure for the training error, which is usually chosen to be the sum of squared errors, previously used for linear regression and which is defined for the ANN as

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{d\in D}(t_d - o_d)^2 \tag{3.14}$$

where d an example belonging to the training set D, being E represented as a function of the weight vector $\boldsymbol{w}$ due to the fact that the outputs o are a function of this same vector, depending on the training set as well.

Taking into account the definition of $E(\boldsymbol{w})$ and visualizing the space composed by two weights w_0 and w_1 and their respective errors E, a parabolic surface comprising a single global minimum is obtained. The weight vector to which the global minimum corresponds can be determined from an initial random weight vector and considering an iterative process which follows the steepest descent direction until the minimum value of E is obtained, in a process known as gradient descent.

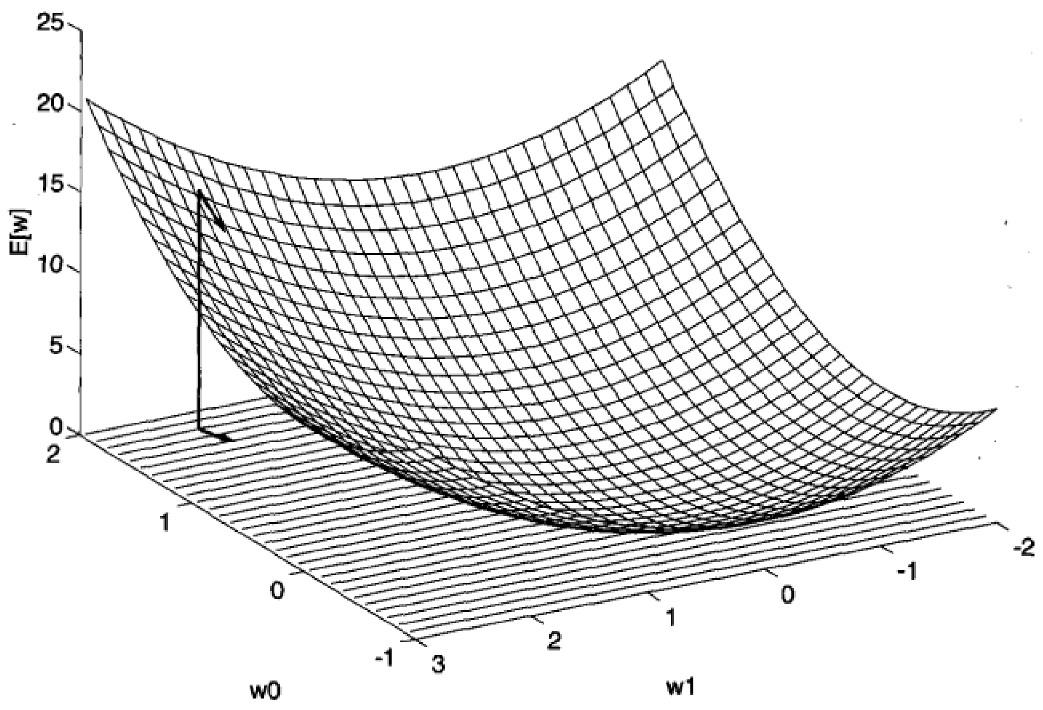


Figure 3.18 – Error surface (Mitchell, T. M. (1997))

Due to the limited nature of perceptron networks which are only capable of representing linear decision boundaries, a new approach known as multilayer networks, capable of representing highly nonlinear decision boundaries, was developed. Multilayer networks work with a continuous, differentiable function, known as sigmoid function, previously seen for the case of logistic regression, allowing for the use of the gradient descent technique.

The consideration of additional layers, known as hidden layers, allows the obtainment of an enlarged space of hypothesis. Despite this however, the ideal number of hidden layers to be considered for the different problems remains a topic under investigation.

Unlike single layered networks, in which the error may be determined by comparing the obtained and the target outputs, in multi-layered networks, this process is not directly applicable since it is not possible to know the target for the hidden layers, making it difficult to determine the error and hence the different weights.

In order to overcome this difficulty, a new algorithm, known as Backpropagation algorithm was developed. This technique is based upon a redefinition of the error E applied in this case to the entirety of the output units and defined as,

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2 \tag{3.15}$$

with t and o, respective the target and obtained outputs for the kth output unit and for the dth example.

The Backpropagation algorithm starts by defining the number of input units, hidden layers and output units and considering random values for the initial configuration of weights as done for the perceptron network. Then for each training example the inputs are propagated forward through the network until the determination of the different outputs, followed by the computation of their respective errors δ_k

$$\delta_k = o_k(1 - o_k)(t_k - o_k) \tag{3.16}$$

These errors are then propagated backwards along the hidden layers, allowing for the determination of the hidden layer's error δ_h and the updated weights w_{ji}.

$$\delta_h = o_h(1 - o_h) \sum_{k \in outputs} w_{kh}\delta_k \tag{3.17}$$

$$w_{ji} = w_{ji} + \Delta w_{ji} \tag{3.18}$$

$$\Delta w_{ji} = \eta \delta_j x_{ji} \tag{3.19}$$

This iterative process is repeated for the array of different examples that belong to the training set until a specified stop criterion, such as the number of iteration or a threshold for the error is met. This sometimes required that the same training set be presented several times in its entirety to the algorithm, each one of these times comprising what is known as an epoch.

Although it is possible to apply the gradient descent method for the determination of the weights vector $\boldsymbol{w}$, the obtained error surface may comprise not only a single global minimum but instead, multiple local minima. Despite this, the Backpropagation method has allowed for good results in a large array of different practical problems.

3.5.7 Support Vector Machines

A Support Vector Machine (SVM) is a learning algorithm that is inserted in the supervised learning framework, and which is based upon statistical learning techniques. SVM’s were developed in the 1990’s by Vapnik as a binary classification technique, although regression-oriented versions are also available, making them a fairly recent approach when compared to other learning algorithms.
This learning algorithm has some particular characteristics which make it a very robust and high-performance ML technique and hence a rather popular one, being the recognition of handwritten digits a common application. Among these characteristics one can highlight its good generalization capabilities, obtained by maximizing the distance between the example points and the decision boundary, a process known as maximum margin separator; the use of the kernel trick, allowing the algorithm to cope not only with linear separable data but also with nonlinear datasets, giving SVM’s a great advantage when compared to learning techniques which are exclusively oriented to linear representations. Due to the nonparametric nature of this learning algorithms, SVM’s may need to store a significant portion of the presented examples, allowing it the possibility to represent complex functions, this however is counterbalanced by the fact that in reality the algorithm will only use a small part of these examples, the ones it considers more important, hence contributing for a good behaviour regarding the issue of overfitting.

As mentioned above, SVM’s have its roots in the field of statistical learning. Their goal is to minimize expected generalization loss instead of the empirical loss, in other words, instead of attempting to minimize the loss (the number of mislabelled examples in case of classification problems) inferred from the training set, the algorithm tries to minimize the problem’s real loss by assuming that the unseen data has the same distribution as the examples presented in the training set and by looking for the previously mentioned maximum margin separator.

Taking into consideration the simplest SVM’s, those that revolve around binary classification, also known as Support Vector Classification (SVC), in which the goal is to find the best plane that separates two classes, a positive and a negative one, it is easy to verify from Figure 3.19

that different separating planes, known as hyperplanes, can be drawn, however only the maximum margin separator will minimize generalization loss.

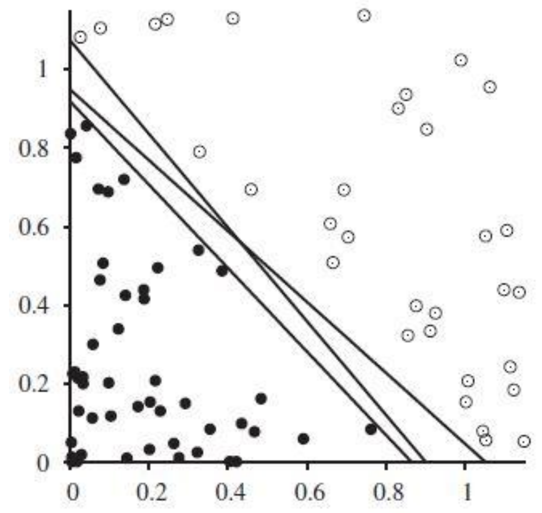


Figure 3.19 – Linear separators (Russel, S. J., Norvig, P. (2016))

A margin can be geometrically interpreted as the width that separates the closest data points of different classes, these particular data points are known as support vectors.

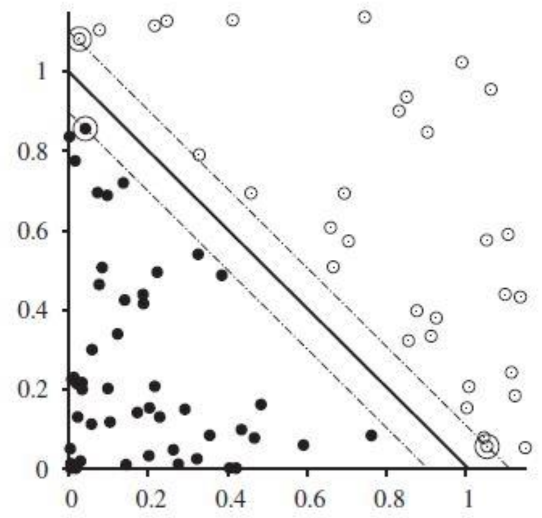


Figure 3.20 – Maximum margin separator and support vectors (Russel, S. J., Norvig, P. (2016))

The maximal margin classifier or hard margin SVM is based upon a linear function in the form of equation (3.20) that assigns a +1 or -1 classification for $f(x) \geq 0$ or $f(x) < 0$ respectively and can only be applied to linearly separable data.

$$f(x) = (\boldsymbol{w}.\boldsymbol{x}) + b \tag{3.20}$$

where $\boldsymbol{w}$ is the weight vector and b is the biases, the former defines a direction perpendicular to the hyperplane while the later moves the hyperplane along its parallel direction.

The maximal margin hyperplane is defined as,

$$(\boldsymbol{w}.\boldsymbol{x}) + b = 0 \tag{3.21}$$

Considering the imposed restrictions

$$(\boldsymbol{w}.\boldsymbol{x}) + b \geq +1, \text{ for } y_i = +1 \tag{3.22}$$

$$(\boldsymbol{w}.\boldsymbol{x}) + b \leq -1, \text{ for } y_i = -1 \tag{3.23}$$

the linear classifiers that separate a set have a positive margin, meaning that the above restrictions ensure that does not exist any example between the hyperplanes defined by both

$$(\boldsymbol{w}.\boldsymbol{x}) + b = 0 \text{ and } |(\boldsymbol{w}.\boldsymbol{x}) + b| = 1 \tag{3.24}$$

Considering that the Euclidian distance between the support vectors and the separating hyperplane can be represented by d_+ and d_- respective for the positive and negative support vectors, and that ρ is the maximum margin obtained between all the hyperplanes and thus $\rho = d_+ + d_-$.

The distance between any given point x_i and an hyperplane $(\boldsymbol{w}, b)$ is given by

$$d_i(\boldsymbol{w}, b, x_i) = \frac{|(\boldsymbol{w}.x_i) + b|}{||\boldsymbol{w}||} = \frac{y_i((\boldsymbol{w}.x_i) + b)}{||\boldsymbol{w}||} \tag{3.25}$$

and as a result,

$$d_i(\boldsymbol{w}, b, x_i) \geq \frac{1}{||\boldsymbol{w}||} \tag{3.26}$$

The above expression leads to $d_+ = d_- = \frac{1}{||\boldsymbol{w}||}$ and thus,

$$\rho = \frac{2}{||\boldsymbol{w}||} \tag{3.27}$$

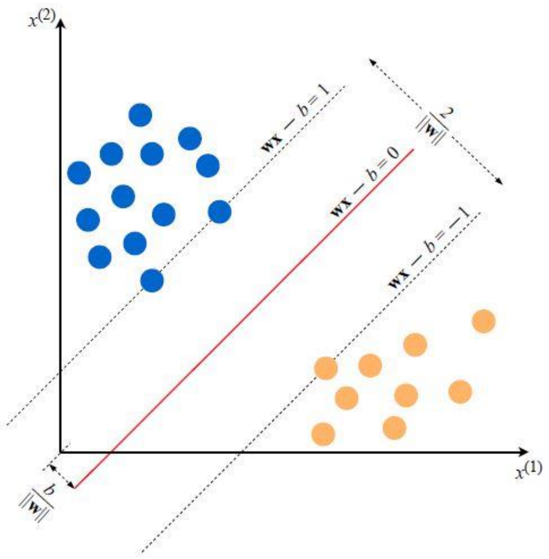


Figure 3.21 – Margin definition (Burkov, 2019)

As can be concluded, the maximum margin separator that will allow the best generalization capabilities corresponds to,

$$max\ \rho \tag{3.28}$$

subject to $y_i((\boldsymbol{w}.x_i) + b) \geq \frac{1}{\|\boldsymbol{w}\|}$, $i = \{1,2, \dots, n\}$ which constitutes the primal problem.

However, this problem is usually rephrased to the equivalent dual problem,

$$min\ \|\boldsymbol{w}\| \tag{3.29}$$

subject to $y_i((\boldsymbol{w}.x_i) + b) \geq 1$, $i = \{1,2, \dots, n\}$

The above mentioned dual problem corresponds to a quadratic programming optimization problem, which results in the definition of the support vectors and the bias b, thus allowing for the classification of a test example through the sign of the function,

$$f(\boldsymbol{z}) = (\boldsymbol{w}^*.\boldsymbol{z}) + b^* \tag{3.30}$$

where the $\boldsymbol{w}^*$ and b^* are the result of the optimization problem.

The classification is obtained simply by means of the dot product between the test set $\boldsymbol{z}$ and the different obtained support vectors.
In most of the practical cases, the structure of the available dataset is noisy hence not linearly separable, and thus, rigid margin SVMs are not able to return a solution.

Unlike hard margin SVMs, soft margin SVMs allow for examples to fall on the wrong side of the separating hyperplane and despite the possibility of misclassified examples, one tries to keep the probability of their existence to a minimum. It may happen as well that although the example is correctly classified, it lies within the margin.

In order to take into account, the possibility of misclassified examples, the previously stated quadratic programming optimization problem is transformed to the form,

$$min\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i \tag{3.31}$$

subject to $y_i((\boldsymbol{w}.x_i) + b) \geq 1 - \xi_i, \xi_i \geq 0, i = \{1,2, \dots, n\}$

where $\sum_{i=1}^{n}\xi_i$ is the soft error and ξ_i are known as slack variables which can be geometrically interpreted as the distance between the misclassified examples and the separating hyperplane. If $\xi_i = 0$ then the example is correctly classified and not within the margin, if $0 < \xi_i < 1$ the example is correctly classified but it falls within the margin and if $\xi_i \geq 1$ then the example is misclassified.
C is a parameter that controls the trade-off between the complexity of the algorithm and the number of misclassified examples., the bigger the value of C, the bigger the penalty when an example is misclassified.

Both hard and soft margin SVMs are linear classifiers, which do not suffice in highly nonlinear datasets. In these cases, a different approach, where the original features are transformed into a higher dimensionality space, in which they can then be linearly separated, may be followed. Figure 3.22 shows a nonlinearly separable dataset in a 2D space transformed into a 3D space in which it is now possible to linearly separate the data.

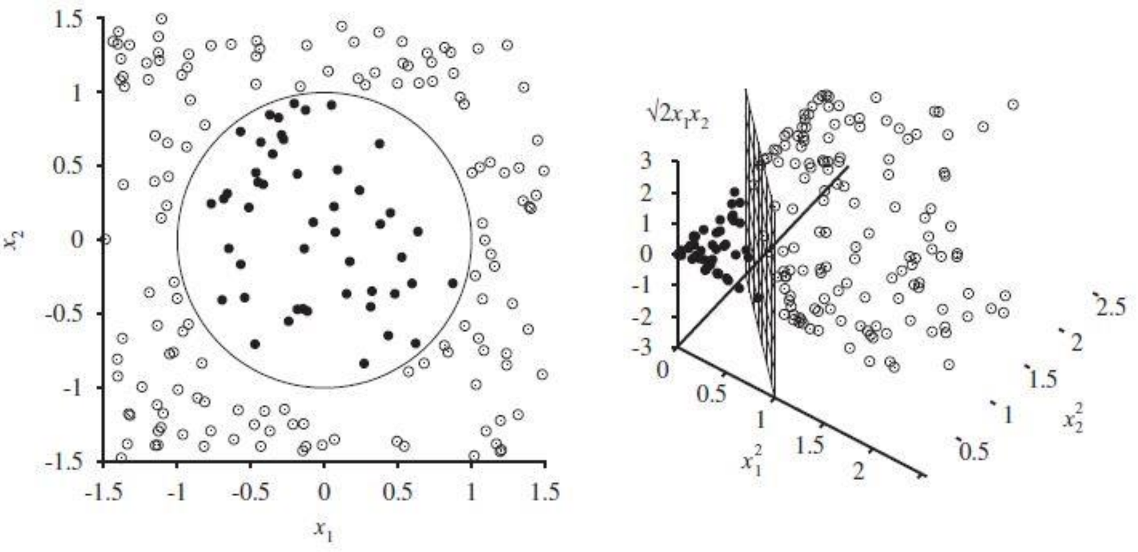


Figure 3.22 – Representation of a single boundary in different dimensions (Russel, S. J., Norvig, P. (2016))

In this approach the increase in dimensionality of the feature vector $\boldsymbol{x}$ is achieved by means of basis functions $\phi(\boldsymbol{x})$. Since the function $\phi(\boldsymbol{x})$ that works best for the dataset in hands is unknown at the start, it would be necessary to transform each example into the higher dimensional space, and then applying the SVM to the data considering different mapping functions, which would become a very inefficient process since first one would have to transform the feature vectors and then proceed to the computation of their dot product. In order to avoid this issue, and since only the previously mentioned dot product is needed, the well-known Kernel Trick is applied, making it possible to compute this dot product after mapping the feature vectors which can be achieved by means of kernel functions. Considering two examples $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, the kernel function can be defined as,

$$\phi(\boldsymbol{x}_1)\ \phi(\boldsymbol{x}_2) = K(\boldsymbol{x}_1, \boldsymbol{x}_2) \tag{3.32}$$

where depending upon the chosen kernel function $K(\boldsymbol{x}_1, \boldsymbol{x}_2)$, SVM can learn among others, d^{th} polynomial classifiers, radial basis (RBF) or sigmoid neural networks.

$$K_{poly}(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\boldsymbol{x}_1 . \boldsymbol{x}_2 + 1)^d \tag{3.33}$$

$$K_{RBF}(\boldsymbol{x}_1, \boldsymbol{x}_2) = e^{(-\gamma(x_1 - x_2)^2)} \tag{3.34}$$

$$K_{sigmoid}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \tanh(k_1(\boldsymbol{x}_1 . \boldsymbol{x}_2) + k_2) \tag{3.35}$$

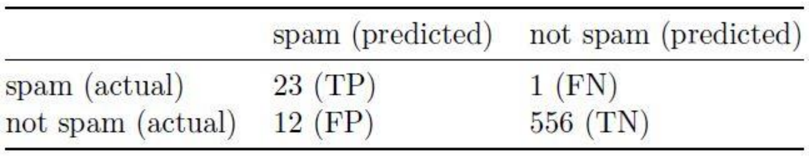
3.6 Evaluation metrics

The ability of a machine learning algorithm to perform a certain task T can be assessed by means of specific quantitative measures of performance P, which allow the user to evaluate the quality of the chosen model (Goodfellow, I., Bengio Y., Courville, A. (2016)).
Depending on the type of problem at hand (classification or regression), different performance assessment metrics can be considered, some of them however may be applied to both problems (Zheng, A., Casari, A. (2018)).

3.6.1 Confusion Matrix

Confusion Matrices are used in Classification tasks and allow, when compared with other metrics, a more detailed analysis of the predicted classification for the different examples and their distribution among the existing classes. The importance in knowing this distribution is related to the cost of misclassifications which can be different depending on the class. This issue can be understood considering an analogous situation such as the one in which a doctor as to diagnose a cancer patient. In this particular situation, the cost of diagnosing the patient with cancer when in reality it does not exist, also known as a False Positive (FP), is rather different from the cost of diagnosing a patient as cancer-free when in reality he or she is not.
Confusion Matrices are not limited to binary classification problems but can also be applied to multiclass ones.

A common example used to present the Confusion Matrix is the classification of emails as "Spam or "Not Spam". In this problem, the learning algorithm is provided with the set of email's text and metadata as input in order to obtain a label as output with classes "Spam" or "Not Spam", which can also be respectively named as "positive" or "negative" or even as "1" or "0".



	spam (predicted)	not spam (predicted)
spam (actual)	23 (TP)	1 (FN)
not spam (actual)	12 (FP)	556 (TN)

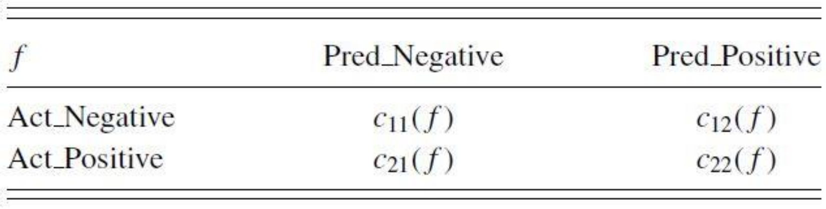
Table 3.1 – Email classification Confusion Matrix (Burkov, A. (2019))

From the analysis of the table above, it is possible to conclude that 23 examples where correctly classified as being spam and are thus True Positives (TP). However, there was 1 example which was classified as not being spam when in reality it in fact was, corresponding to what is known as a False Negative (FN). From the 568 examples that were in reality not spam, 556 examples where correctly classified (True Negatives (TN)), while 12 were wrongly labelled as spam (FP).

Considering the simplest of cases, the Confusion Matrix of a fixed classifier f and a test set of T examples can be mathematically defined as,

$$\boldsymbol{C}(\boldsymbol{f}) = \{c_{ij}(f)\}, i, j \in \{1,2,..,l\} = \sum_{x=1}^{T}[(y = i)\wedge(f(x) = j)] \tag{3.36}$$

where x is a test set example and y the corresponding label. i and j are, respectively, the rows and columns of the matrix and l is the number of classes.
Each element of the matrix $c_{ij}(f)$, corresponds to the number of examples that in reality belong to the class i but were assigned by the classifier the class j.



f	Pred_Negative	Pred_Positive
Act_Negative	$c_{11}(f)$	$c_{12}(f)$
Act_Positive	$c_{21}(f)$	$c_{22}(f)$

Table 3.2 – Generic Confusion Matrix (Japkowicz, N., Shah, M. (2011))

From the analysis of equation (3.36) and Figure 3.24, one can conclude that the diagonal entries c_{ii} of a Confusion Matrix correspond to the correct classified examples, while the remain nondiagonal ones are misclassified examples. Furthermore, the sum of the examples along a row i represent the total number of examples which in reality are labelled as class i while the sum of examples along a column j represent the total number of examples that were predicted to be of class j.

Usually, a Confusion Matrix $\boldsymbol{C}(f)$ measures the performance of a single model f obtained from a fixed learning algorithm. However, when only a small dataset is available it may be necessary to resample the available data, creating multiple pairs of training and test sets, in which case the Confusion Matrix would represent the combined performance over the different pairs. It is also possible for the different entries of the Confusion Matrix to represent not only the performance over the different test sets but also over different models obtained from different learning algorithms.

3.6.2 Error Rate

The Error Rate $R_T(f)$ is a performance metric that returns the ratio of misclassified examples to the complete set of examples comprised in a dataset.

$$R_T(f) = \frac{1}{T}\sum_{i=1}^{T} I(y_i \neq f(x_i)) \tag{3.37}$$

where the indicator function $I(a)$ returns a value of 1 if the condition a is true, and 0 when this is not the case.

Considering the previous definition of the Confusion Matrix, expression (3.37) can also be written in the form of equation (3.38) for the more general case of multiclass tasks and in the form of equation (3.39) for binary classification problems.

$$R_T(f) = \frac{\sum_{i=1}^{l}\sum_{j=1;i\neq j}^{l} c_{ij}(f)}{\sum_{i=1}^{l}\sum_{j=1}^{l} c_{ij}(f)} = \frac{\sum_{i=1}^{l}\sum_{j=1}^{l} c_{ij}(f) - \sum_{i=1}^{l} c_{ii}(f)}{\sum_{i=1}^{l}\sum_{j=1}^{l} c_{ij}(f)} \tag{3.38}$$

$$R_T(f) = \frac{FN+FP}{TP+TN+FP+FN} \tag{3.39}$$

3.6.3 Accuracy

The accuracy $Acc_T(f)$ is the ratio of correctly labelled examples, positive and negative, to the total amount of available examples, and can thus be understood as the complementary of the Error Rate.

$$Acc_T(f) = \frac{1}{T}\sum_{i=1}^{T} I(y_i = f(x_i)) = \frac{\sum_{i=1}^{l} c_{ii}(f)}{\sum_{i=1}^{l}\sum_{j=1}^{l} c_{ij}(f)} \tag{3.40}$$

$$Acc_T(f) = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.41}$$

Together with the Error Rate, Accuracy gives a perspective of the overall performance of the model being evaluated, considering the full range of classes involved in the problem. These however don’t reflect the importance that particular classes may have in the task being undertaken. Moreover, the two above mentioned metrics give better information when the dataset has a balanced distribution among the different classes, opposed to when some particular classes are represented by a considerably larger number of examples, in which case, a biased result, influenced by the more-prevalent class is obtained.

3.6.4 Precision

Precision $Prec_i(f)$ or Positive Predictive Value $PPV_i(f)$ is a metric that can be understood as the ratio of examples that were correctly labelled as being of class i to the total amount of examples which were classified as being of this particular class, and thus, Precision enables the comprehension of how precise the model is in classifying the examples of a given class.

For multiclass tasks this can be represented as,

$$Prec_i(f) = PPV_i(f) = \frac{c_{ii}(f)}{\sum_{j=1}^{l} c_{ji}(f)} \tag{3.42}$$

while for the particular case of binary classification tasks,

$$Prec(f) = PPV(f) = \frac{TP}{TP+FP} \tag{3.43}$$

3.6.5 Recall

Recall, usually known as Sensitivity or True Positive Rate $TPR_i(f)$, returns the ratio of the examples that were labelled by the classifier as being of class i to the total amount of examples which actually belong to the class being surveyed.

$$Rec_i(f) = TPR_i(f) = \frac{c_{ii}(f)}{\sum_{j=1}^{l} c_{ij}(f)} \tag{3.44}$$

$$Rec(f) = TPR(f) = \frac{TP}{TP+FN} \tag{3.45}$$

3.6.6 F measures

The different F measures, combine both Precision and Recall into a single metric by means of a weighted harmonic mean which in its most general form can be represented as,

$$F_\beta = \frac{(1+\beta^2)(Prec(f)*Rec(f))}{(\beta^2*Prec(f))+Rec(f)} \tag{3.46}$$

where $\beta \in \mathbb{R}$ with $\beta > 0$.

The F1 score, with $\beta = 1$, takes into account both Precision and Recall evenly (3.47), while the F2 score for its turn doubles the weight of recall in comparison to precision (3.48).

$$F_1 = \frac{2(Prec(f)*Rec(f))}{Prec(f)+Rec(f)} \tag{3.47}$$

$$F_2 = \frac{5(Prec(f)*Rec(f))}{(4*Prec(f))+Rec(f)} \tag{3.48}$$

3.6.7 Root-mean-squared-error

The root-mean-squared-error (RMSE) is the most widely used metric for the assessment of regression tasks and can be defined as the root of the averaged squared distance between the predicted value and the actual value.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - f(x_i))^2}{n}} \tag{3.49}$$

where y_i is the actual value of the example, $f(x_i)$ is the corresponding predicted value and n is the total number of available examples.

3.6.8 Mean absolute percentage error

The mean absolute percentage error (MAPE) is obtained by first determining the error of the predicted value in comparison to the actual value. The error is then transformed into a percentage of the actual value and an average of the errors of the various examples is then obtained and can be represented by equation 3.50.

$$MAPE = \frac{1}{n}\sum_{i=0}^{n-1}\frac{|y_i - f(x_i)|}{|y_i|} \tag{3.50}$$

3.7 Summary

In this Chapter were presented the most common types of problems and the different algorithms frequently used for their resolution as well as some of the evaluation metrics available for the analysis of the suitability of the created models. The application of these techniques to the proposed problem as well as the evaluation of the obtained results is presented in the following Chapters.

4 Proposed Approach

This chapter presents the proposed approach to tackle the problem of designing unreinforced welded beam to column connections using machine learning techniques. As mentioned previously in chapter 1, this problem will be divided into two different problems. A classification problem to identify the conditioning component and a regression problem to obtain the corresponding design resistant bending moment.

Considering the current procedure followed for the design of steel joints, the need to repeat this same process for a large set of similar connections within a single project as well as the large number of existing data related to previously validated examples, makes the application of machine learning techniques an interesting approach to transform this procedure into an easier and faster one.

The application of these methods to the design of steel connections and the corresponding validation of its results make the obtention of a dataset, as well as the development of different models an essential part of this work.

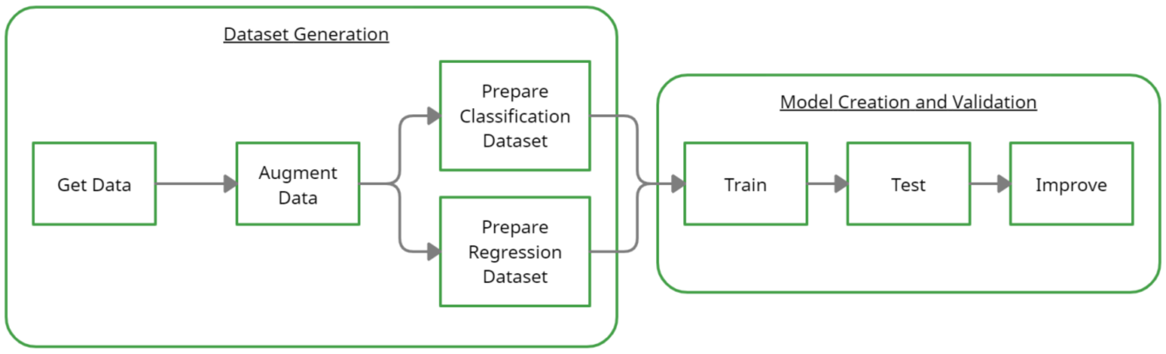

Dataset Generation
Get Data
Augment Data
Prepare Classification Dataset
Prepare Regression Dataset
Model Creation and Validation
Train
Test
Improve

Figure 4.1 – Proposed workflow

The workflow used for the development of the proposed work presented on Figure 4.1 was based upon the Machine Learning workflow of Figure 3.3.

The process of creating and augmenting the dataset as well as its division into two different sets used for the two proposed problems (Classification and Regression) are fitted into Chapter 4.1 “Dataset Generation”. The procedures followed to train the different models and the analysis of the obtained results based upon the respective tests are presented in Section 4.2 and Chapter 5.

4.1 Dataset generation

The development of the proposed work depends necessarily on the creation of a dataset composed of a significant volume of examples that are representative of the type of unreinforced welded connections that practitioners are faced with in their work. Taking into account the difficulty in constructing a dataset with real examples developed during the design of buildings or other structures, namely due to the need of consent to use this data from the different parts involved for each of the different projects, a different approach was followed. This approach was implemented through the development of a simple script, using everyday tools such as Excel and Visual Basic for Applications (VBA), for the design of welded beam to column connections accordingly to EN 1993-1-1 and EN 1993-1-8, considering a wide range of European I and H sections and.

Considering that one of the objectives is to construct the dataset to represent the widest range of I or H sections, including welded built-up sections, its development was made considering that the inputs to be fed to the different learning algorithms should comprise geometric characteristics that are not only analogous among the different sections but that also translate in some degree their resistance. Thus, a total of 8 different geometric characteristics, represented as real-valued numbers, for both the beam and column were considered.

- h_b the height of the beam
- b_b the width of the beam
- t_{wb} the thickness of the beam’s web
- t_{fb} the thickness of the beam’s flange
- h_c the height of the column
- b_c the width of the column
- t_{wc} the thickness of the column’s web
- t_{fc} the thickness of the column’s flange

In addition to the 8 features comprising each example, 2 different outputs were considered. The first, to be used in classification problems, corresponds to the governing component, the one that limits the connection’s resistance for the different examples. Taking into account the interest in determining the resistance of an unreinforced welded connection given the sections of the beam and column, a second output, consisting of a real-valued number, which represents the resistance of the connection due to its weakest component was considered, allowing for the development of regression analysis. Both outputs were obtained considering the same S355 steel grade for both the column and beam elements.

Due to the different nature of both outputs, the original dataset was split into two distinct datasets, both with the same number of examples and features but each with its corresponding output.

The flowchart represented in Figure 4.1 shows the algorithm developed and implemented for the creation of the dataset.
The first step consists in assigning a section for the beam b_i to be connected which can be characterized by the dimensions h_{bi}, b_{bi}, t_{wbi} and t_{fbi}. Once the beam section has been defined, one proceeds to the definition of the column's section c_j which is characterized by its respective dimensions h_{cj}, b_{cj}, t_{wcj} and t_{fcj}.

For each pair beam-column, a set of limits are considered for the ratios h_{bi}/h_{cj} and b_{bi}/b_{cj}, in order to avoid unrealistic beam-column pairs. The assumed limitations are as follows.

- $b_{bi} \leq b_{cj}$
- $b_{bi} \geq 0.5 b_{cj}$
- $h_{bi} \leq 2 h_{cj}$
- $h_{bi} \geq 0.5 h_{cj}$

In case any of the above-mentioned conditions are not met, the pair in analysis is discarded and the following column section is considered. If all c_j sections were previously considered, a new cycle begins with a new beam section.
Otherwise, if all the conditions are respected, the algorithm assigns the null value to the design bending moment acting on the beam $M_{Ed,b} = 0$, and an increment of bending moment $M_{Rd,step}$ corresponding to 1% of the smaller resistant bending moment between the beam and the column sections is considered $M_{Rd,min} = \min(M_{Rd,bi}; M_{Rd,cj})$. The design bending moment $M_{Ed,b}$ is then successively incremented, as well as the bending moment acting in the column above and below the beam which as a simplification, is assumed to be $M_{Ed,c} = \frac{M_{Ed,b}}{2}$.

In each cycle n for which the design bending moments are incremented, the algorithm engages in an analysis of the utilization ratio R_n of the different components of the connection, except for the flange welds component, as it is assumed that these are full strength and thus do not govern the resistance of the connection. The cycle ends when the ratio $R_n \geq 1$, in which case the governing component is determined as well as its corresponding resistance in the form of the maximum bending moment that may act on the beam, $M_{Rd,b} = \frac{M_{Ed,b\,n}}{R_n}$.

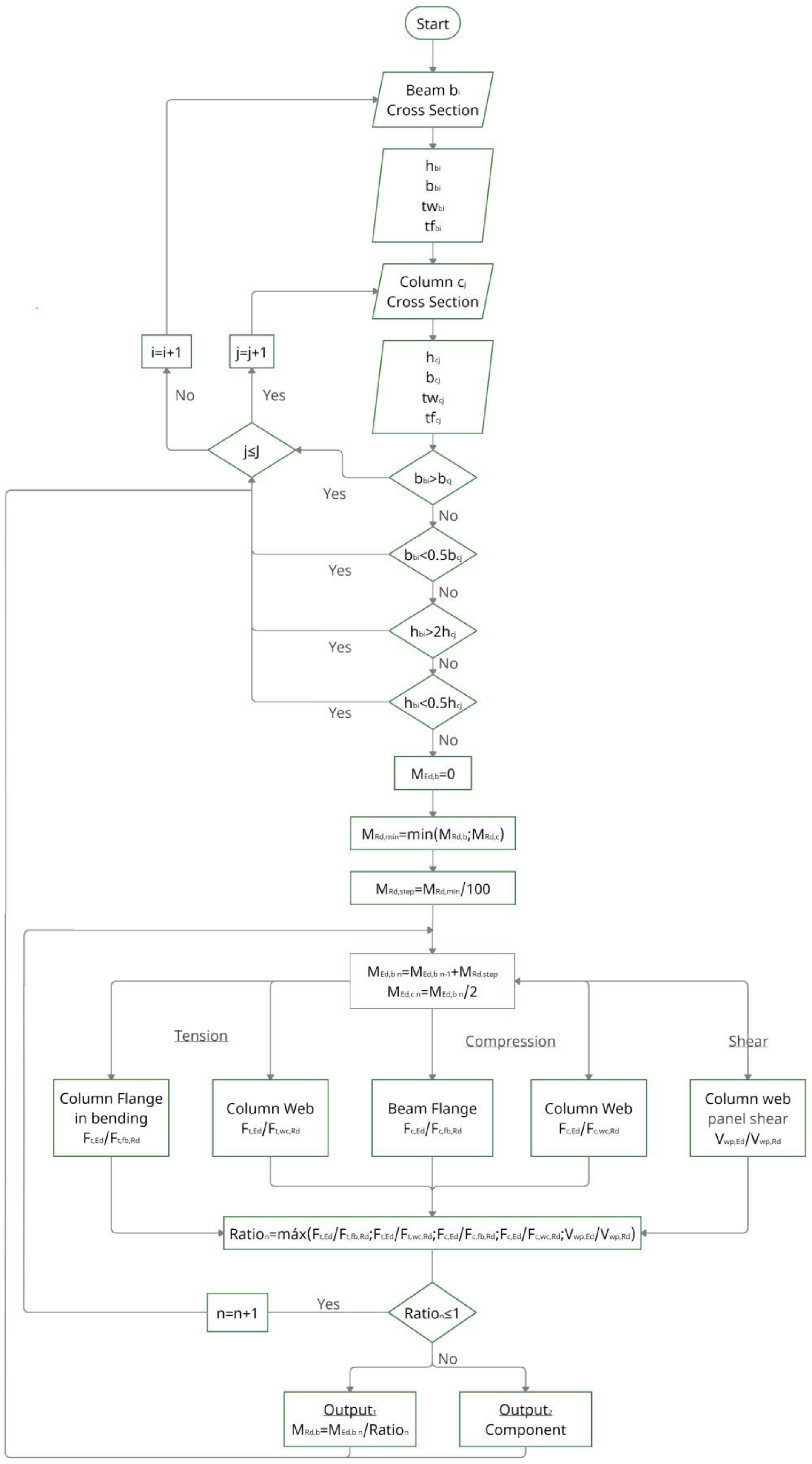

Start
Beam b_i Cross Section
h_{bi}
b_{bi}
tw_{bi}
tf_{bi}
Column c_j Cross Section
h_{cj}
b_{cj}
tw_{cj}
tf_{cj}
i=i+1
j=j+1
No
Yes
j≤J
$b_{bi}>b_{cj}$
Yes
No
$b_{bi}<0.5b_{cj}$
Yes
No
$h_{bi}>2h_{cj}$
Yes
No
$h_{bi}<0.5h_{cj}$
Yes
No
$M_{Ed,b}=0$
$M_{Rd,min}=min(M_{Rd,b};M_{Rd,c})$
$M_{Rd,step}=M_{Rd,min}/100$
$M_{Ed,b\ n}=M_{Ed,b\ n-1}+M_{Rd,step}$
$M_{Ed,c\ n}=M_{Ed,b\ n}/2$
Tension
Compression
Shear
Column Flange in bending $F_{t,Ed}/F_{t,fb,Rd}$
Column Web $F_{t,Ed}/F_{t,wc,Rd}$
Beam Flange $F_{c,Ed}/F_{c,fb,Rd}$
Column Web $F_{c,Ed}/F_{c,wc,Rd}$
Column web panel shear $V_{wp,Ed}/V_{wp,Rd}$
$Ratio_n=máx(F_{t,Ed}/F_{t,fb,Rd};F_{t,Ed}/F_{t,wc,Rd};F_{c,Ed}/F_{c,fb,Rd};F_{c,Ed}/F_{c,wc,Rd};V_{wp,Ed}/V_{wp,Rd})$
n=n+1
Yes
$Ratio_n\leq1$
No
$Output_1$ $M_{Rd,b}=M_{Ed,b\ n}/Ratio_n$
$Output_2$ Component

Figure 4.2 – Dataset generation flowchart – 9079 datapoints

This process is then repeated for the different beam-column pairs, allowing for a total of 9079 different combinations. Figure 4.2 represents the distribution of the obtained beam-column pairs in a 2D plot with the beam height h_b in the abscissa and the column height h_c in the ordinate axis. From the analysis of this plot one can highlight the clear distinction of the upper and lower limits introduced by the constraints $h_{bi} \leq 2h_{cj}$ and $h_{bi} \geq 0.5h_{cj}$. Furthermore, the dataset as the appearance of an artificially created one, reflected mainly by the organized distribution of the examples.

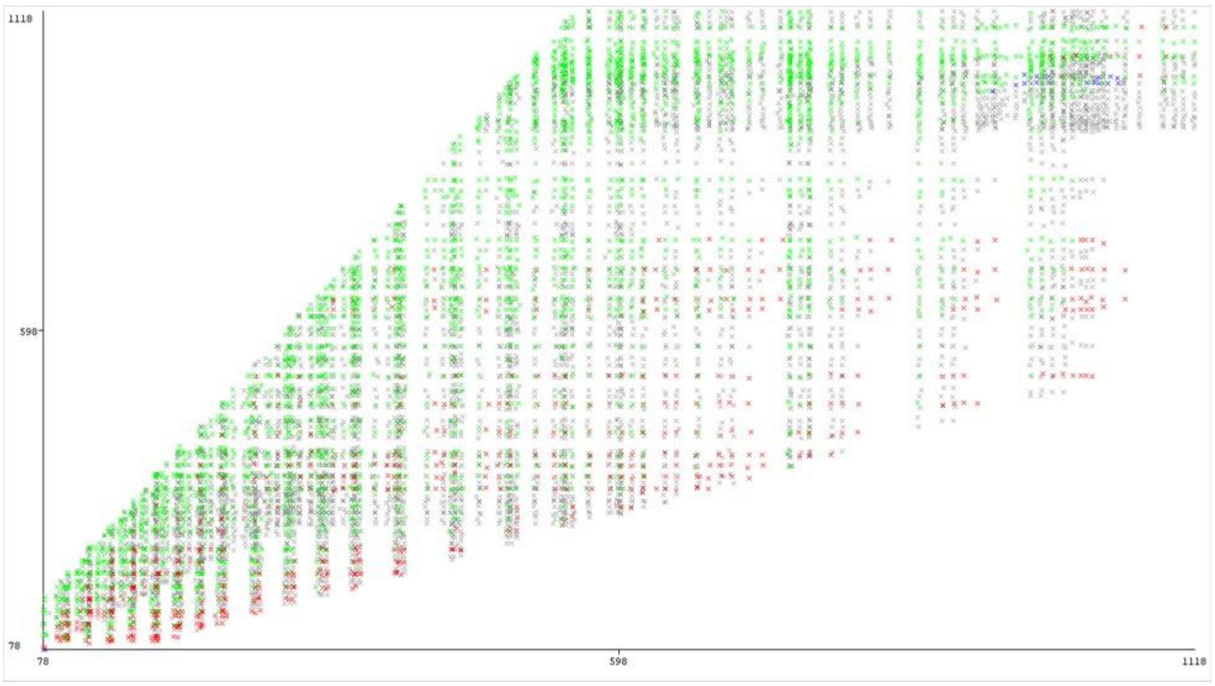


Figure 4.3 – Dataset visualization – Beam height vs Column height

In order to increase the number of data points comprising the dataset and giving it a more disperse and realistic distribution of examples, a new set of 10 beam-column pairs with dimensions h^*, b^*, tw^* and tf^* was obtained from each one of the previous 9079 combinations as represented on the flowchart of Figure 4.3. This procedure was based upon the ratio of the mean to nominal dimensions of the different sections and their corresponding coefficient of variation (c.o.v), which by considering a normal distribution allowed to obtain a total of 99849 different beam-column configurations distributed among 3 different classes, Column Flange in bending, Compression Web and Web Shear.

Dimension	b	h	t_w	t_f
mean/nominal	1	1	1	0.98
c.o.v	0.9%	0.9%	2.5%	2.5%

Table 4.1 – Standard steel sections dimensional distribution

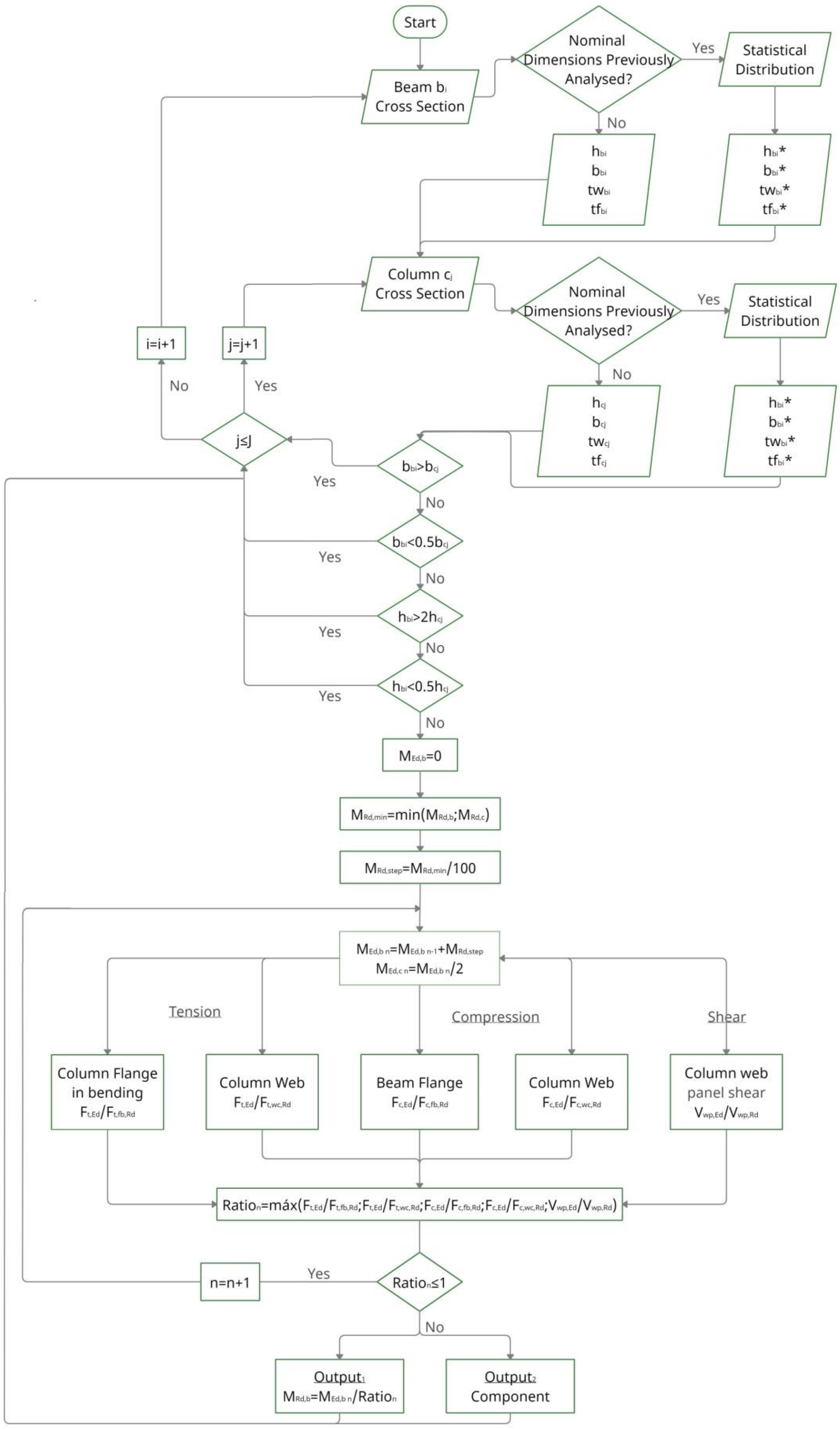

Start
Beam b_i Cross Section
Nominal Dimensions Previously Analysed?
Yes
No
Statistical Distribution
h_{bi} b_{bi} tw_{bi} tf_{bi}
h_{bi}* b_{bi}* tw_{bi}* tf_{bi}*
Column c_j Cross Section
Nominal Dimensions Previously Analysed?
Yes
No
Statistical Distribution
h_{cj} b_{cj} tw_{cj} tf_{cj}
h_{bi}* b_{bi}* tw_{bi}* tf_{bi}*
i=i+1
j=j+1
No
Yes
j≤J
$b_{bi}>b_{cj}$
Yes
No
$b_{bi}<0.5b_{cj}$
Yes
No
$h_{bi}>2h_{cj}$
Yes
No
$h_{bi}<0.5h_{cj}$
Yes
No
$M_{Ed,b}=0$
$M_{Rd,min}=min(M_{Rd,b};M_{Rd,c})$
$M_{Rd,step}=M_{Rd,min}/100$
$M_{Ed,b\,n}=M_{Ed,b\,n-1}+M_{Rd,step}$
$M_{Ed,c\,n}=M_{Ed,b\,n}/2$
Tension
Compression
Shear
Column Flange in bending $F_{t,Ed}/F_{t,fb,Rd}$
Column Web $F_{t,Ed}/F_{t,wc,Rd}$
Beam Flange $F_{c,Ed}/F_{c,fb,Rd}$
Column Web $F_{c,Ed}/F_{c,wc,Rd}$
Column web panel shear $V_{wp,Ed}/V_{wp,Rd}$
$Ratio_n=máx(F_{t,Ed}/F_{t,fb,Rd};F_{t,Ed}/F_{t,wc,Rd};F_{c,Ed}/F_{c,fb,Rd};F_{c,Ed}/F_{c,wc,Rd};V_{wp,Ed}/V_{wp,Rd})$
n=n+1
Yes
$Ratio_n≤1$
No
$Output_1$ $M_{Rd,b}=M_{Ed,b\,n}/Ratio_n$
$Output_2$ Component

Figure 4.4 – Dataset generation flowchart – 99849 data points

From the analysis of Figure 4.4, it is noticeable, the increased number of data points and their more disperse distribution.

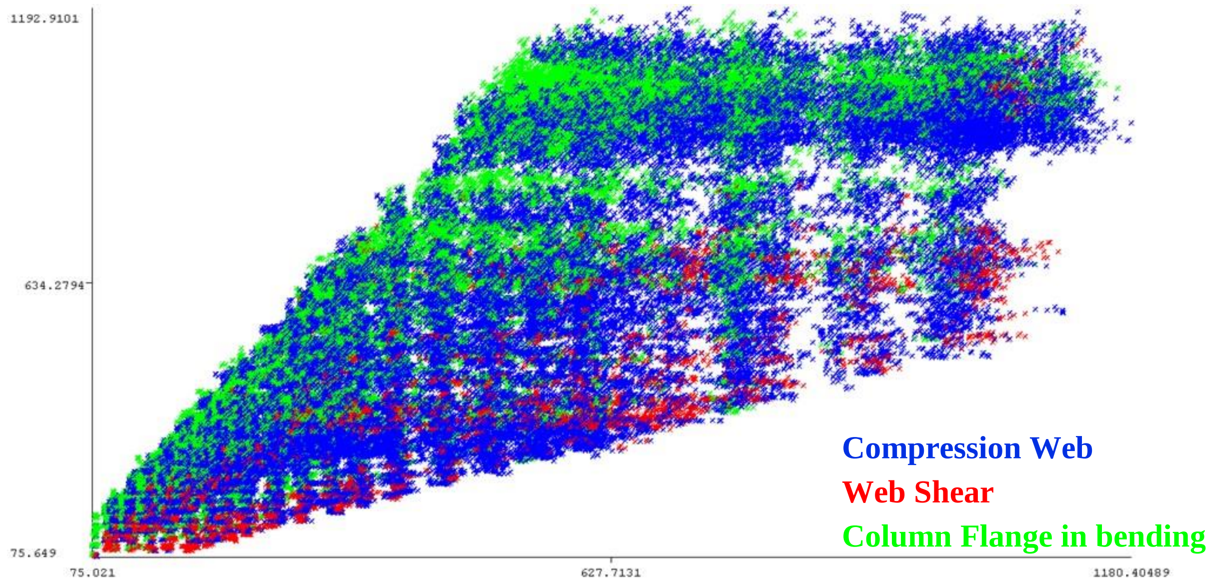


Figure 4.5 – Dataset visualization – Beam height [mm] (abscissa) vs Column height [mm] (ordinate)

4.2 Model creation and validation

Once the dataset has been obtained, it becomes necessary to define the best approach to be followed in order to apply the different learning algorithms. Thus, and taking into account its versatility and its rising popularity amongst the Machine Learning community, programming language Python was chosen to be the default one for the development of this work. Moreover, the different pre-processing operations needed, the different learning algorithms used as well as the different evaluation metrics used for their validation were implemented by means of open source Machine Learning library sklearn[2].

[2] https://scikit-learn.org/

5 Results and Analysis

This chapter is divided in three parts. The first addresses the Exploratory Data Analysis (EDA) which is commonly applied before any learning technique, while the last two correspond to the classification and regression tasks undertaken by applying different learning algorithms, and by varying the corresponding hyperparameters as well as the number of input features.

5.1 Exploratory Data Analysis

The process of Exploratory Data Analysis is usually undertaken for the development of an early analysis of the available dataset, previously to the application of any machine learning technique. This procedure allows the determination of specific characteristics such as patterns or anomalies that may be, at first sight at least, hidden within the dataset and that may be initially disclosed by means of techniques such as the graphic visualization of the dataset.

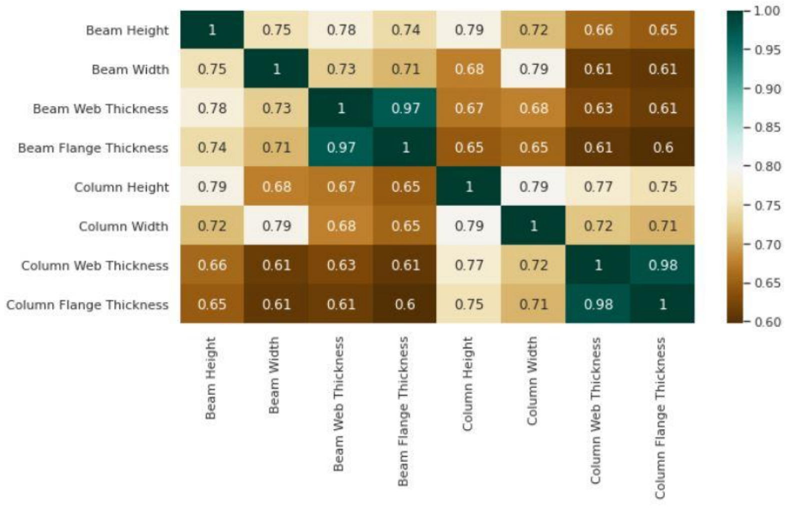


Figure 5.1 – Feature Correlation

The development of this analysis was made by means of different software. Initially, in order to obtain the 2D representations of the dataset, the well-known Weka[3] software was used. The will to further develop this analysis, by obtaining also 3D plots, led to the use of Matlab.

Figure 5.1, obtained by means of the Python programming language run within GoogleColab[4] , presents the correlation matrix of the 8 considered features. As it can be seen, the diagonal elements relate the different features with themselves, leading to a full correlation, with a value of 1.

[3] https://www.cs.waikato.ac.nz/~ml/weka/

[4] https://colab.research.google.com/

It should also be mentioned the high correlation obtained between the web and flange thicknesses of both the columns and the beams, respectively 0.97 and 0.98. The difference between the two values is a result of not considering exclusively the standard dimensions of the different sections but also those that despite being based upon their standard counterparts, were obtained by means of a normal distribution.

The second highest obtained correlation (0.79) refers to the column height and its width. This reduction, when compared to the previously mentioned correlations, is a result of the consideration of sections such as HEA and HEB type sections. The width of these sections increase along with the increase of its height until a certain dimension is reached. Once the threshold dimension is met, the height of section continues to increase while the corresponding width remains constant.

The obtainment of the correlation matrix can be used as a support for the feature engineering process undertaken for the different learning algorithms, enabling an informed choice of the features to be successively discarded.

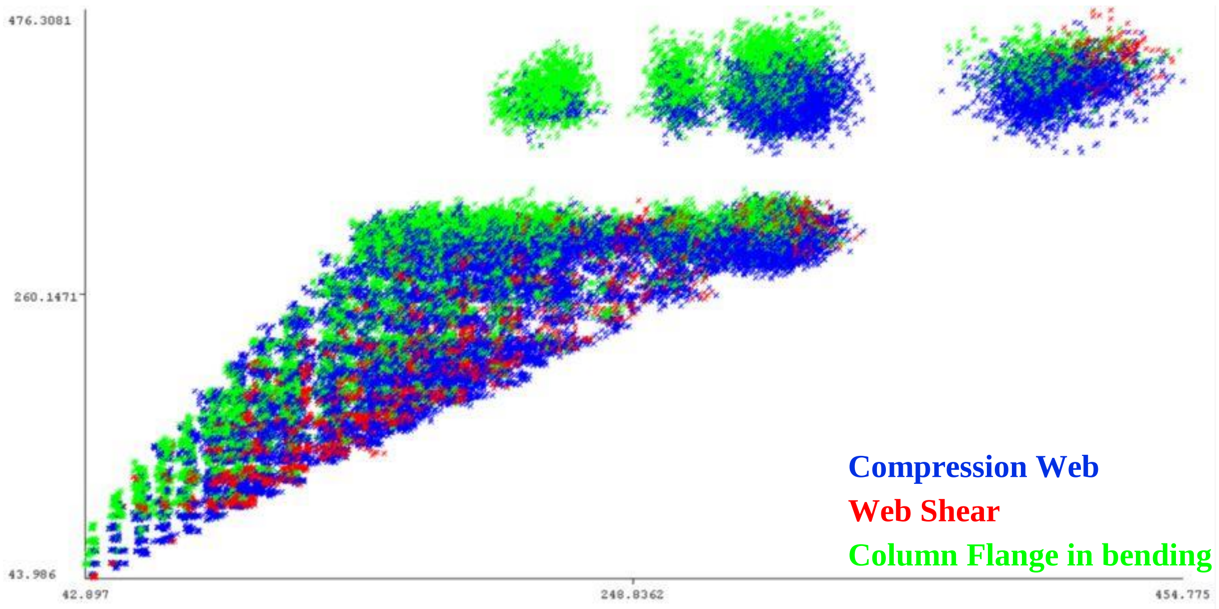


Figure 5.2 – 2D Dataset visualization – Beam width [mm] (abscissa) vs Column width [mm] (ordinate)

The obtainment of relevant results throughout the graphic visualization process depends upon the choice of the different inputs to relate. Figure 5.2 presents the set of data points by means of a 2D plot with the beam width values and the column width in the abscissa and ordinate, respectively. Also, the distribution of the different points with respect to the conditioning component is displayed, blue for the column compression web, green for the column Flange in bending and red for the column web shear. From the analysis of Figure 5.2, it is not possible to extract meaningful information due to the dispersion and superposition of data points of different classes. Despite this, it should be mentioned the graphical visualization of the previously mentioned imposed limits to the h_{bi}/h_{cj} and b_{bi}/b_{cj} ratios and the fact that the considered sections do not possess a continuous range of widths, with the absence of the [315;420]mm range.

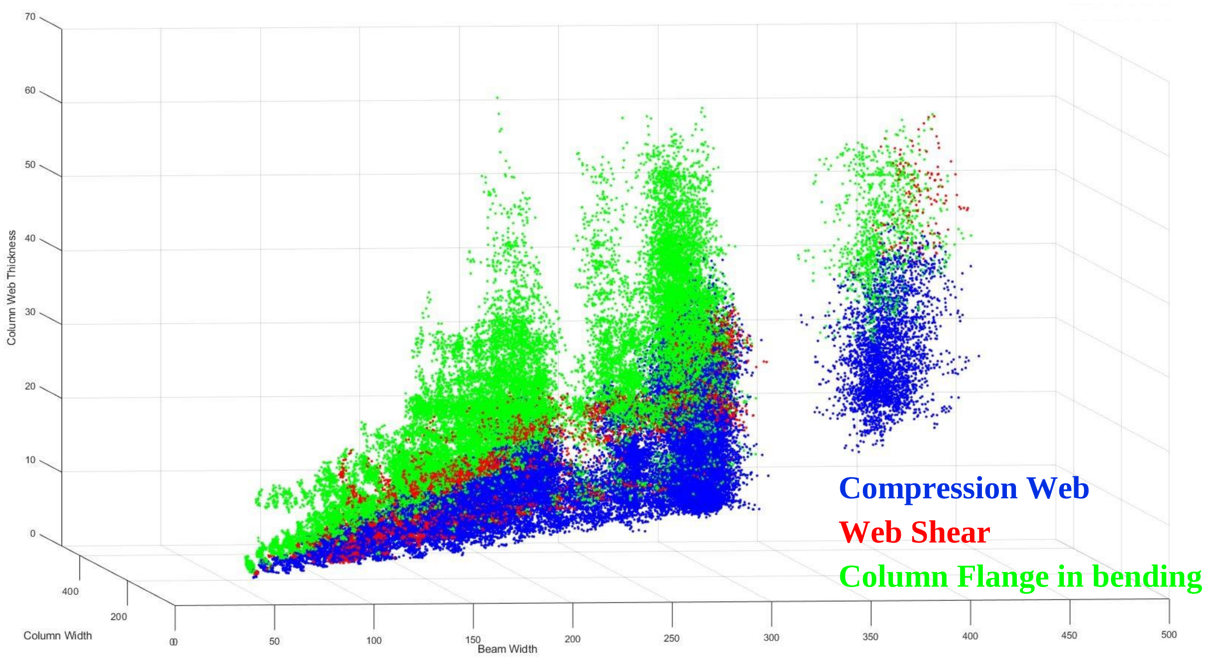


Figure 5.3 – 3D Dataset visualization – Beam width [mm] (abscissa) vs Column width [mm] (ordinate) vs Column web thickness [mm] (applicate)

Using the representation presented in Figure 5.2 as a basis, and considering an additional feature, in this case, the column web thickness, it is possible to obtain the 3D plot of Figure 5.3, in which it is possible to notice not only the same range of missing widths [315,420]mm but also an improvement in the separation between classes, a direct result of considering an extra dimension for the representation of the dataset.

The use of a 3D representation of the dataset allows not only a better general idea of the distribution of classes but also the representation of the distribution of the resistance of the different beam-column pairs as a function of their respective conditioning component, by considering this resistance in the applicate axis.

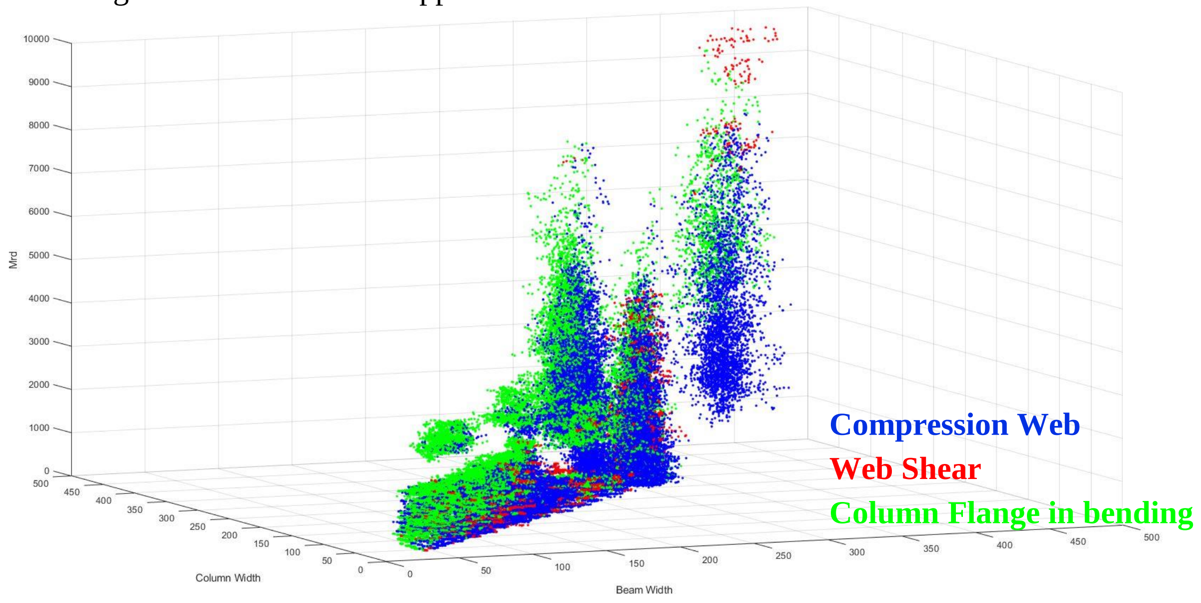


Figure 5.4 – 3D Dataset visualization – Beam width [mm] (abscissa) vs Column width [mm] (ordinate) vs Mrd [kN.m] (applicate)

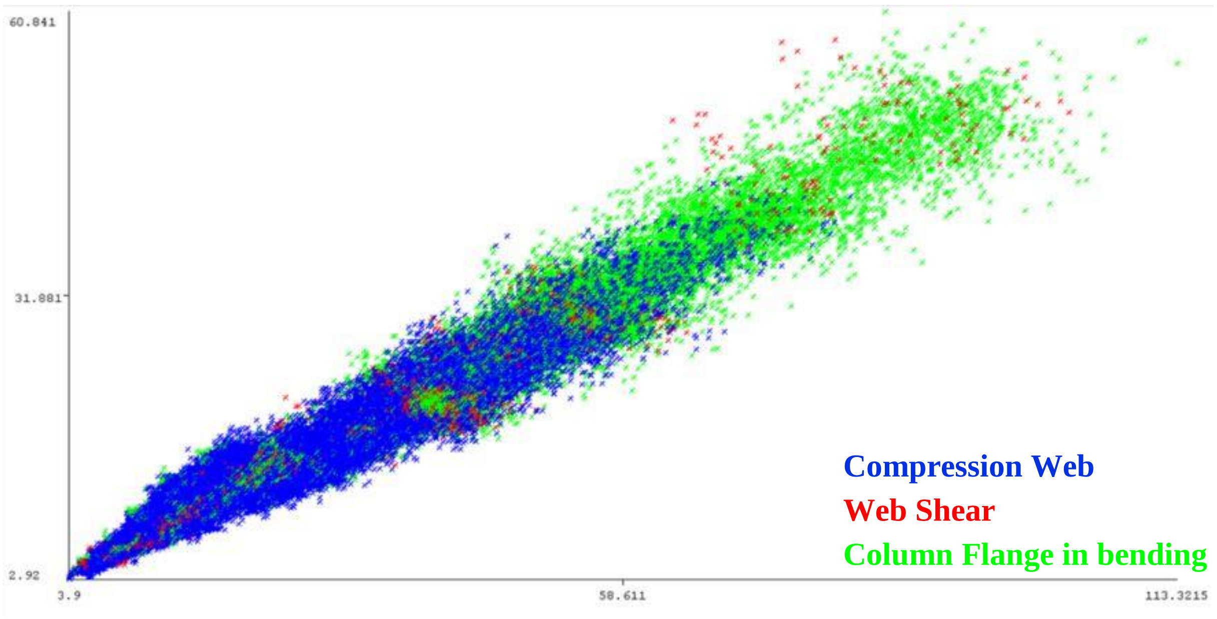


Figure 5.5 – 2D Dataset visualization – Column flange thickness [mm] (abscissa) vs Column web thickness [mm] (ordinate)

Considering the plot presented in Figure 5.5, with the abscissa and ordinate axis representing respectively the column flange thickness and the column web thickness, it is noticeable an improvement translated by a clearer transition zone between two classes, the column compression web and the column flange in bending, while the third class however remains considerably disperse.

The separation between the two previously mentioned classes is a result of the influence of smaller column web thickness in the resistance of the connection, while the increase of this thickness leads to the change of the connection's resistance governing factor and thus to the corresponding conditioning component.

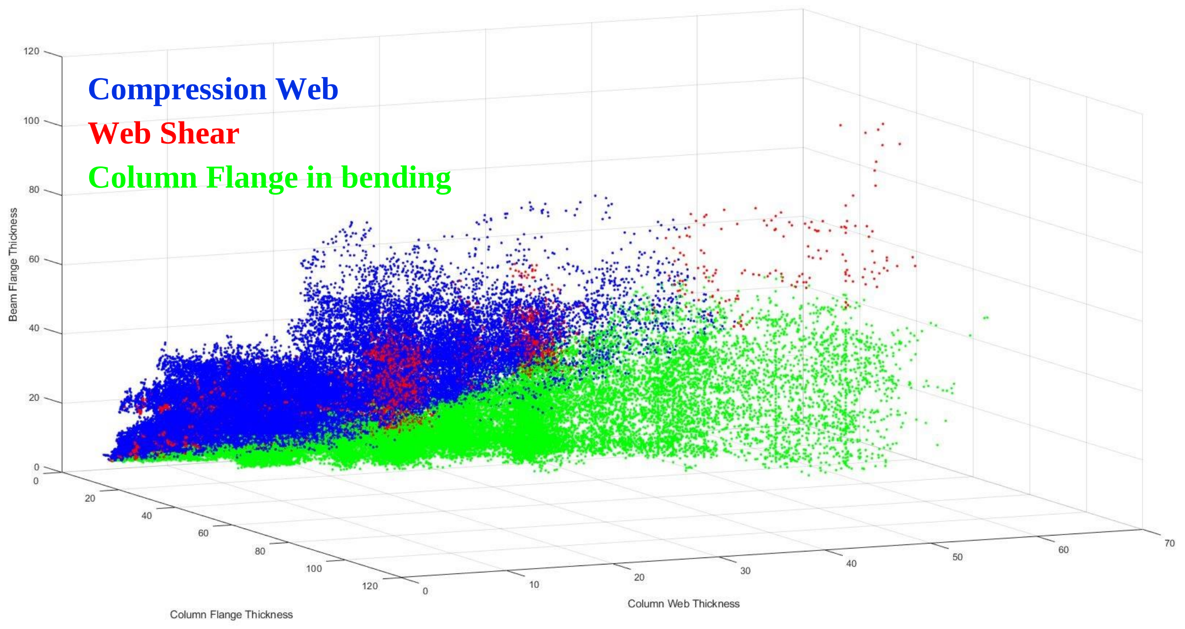


Figure 5.6 – 3D Dataset visualization – Column flange thickness [mm] (abscissa) vs Column web thickness [mm] (ordinate) vs Beam flange thickness [mm] (applicate)

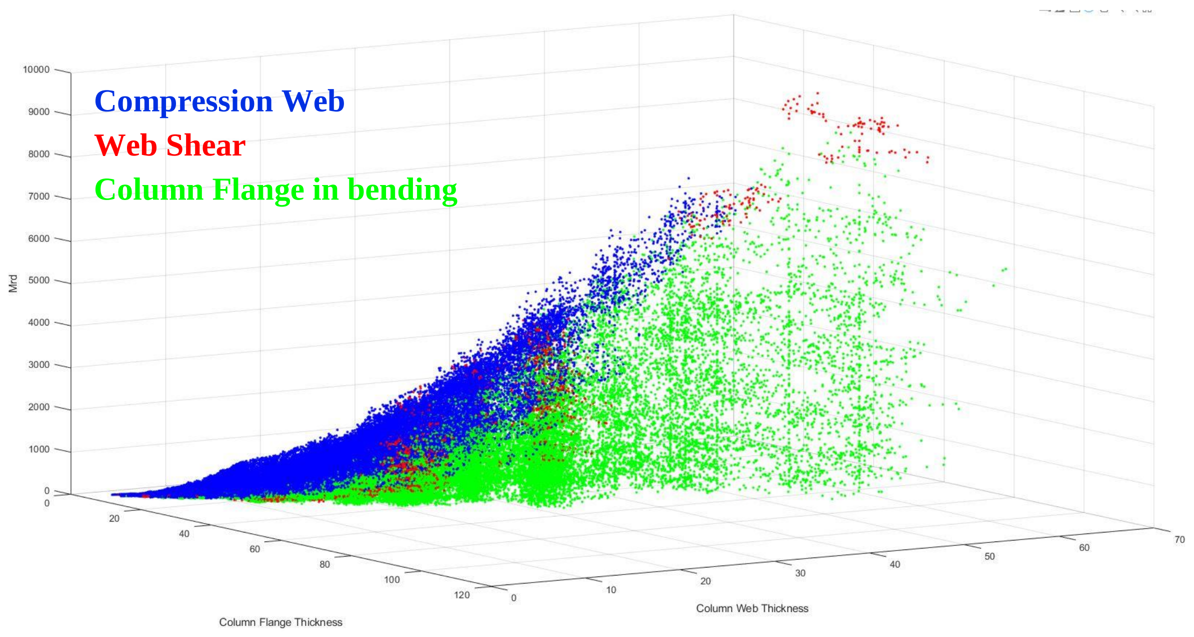


Figure 5.7 – 3D Dataset visualization – Column flange thickness [mm] (abscissa) vs Column web thickness [mm] (ordinate) vs Mrd [kN.m] (applicate)

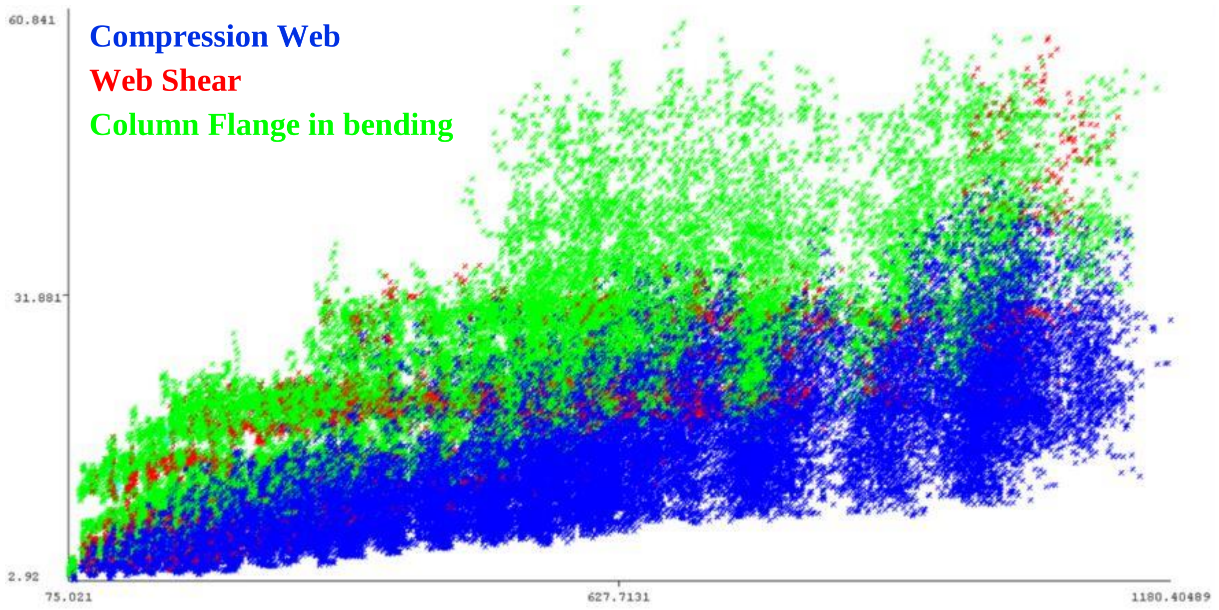


Figure 5.8 – 2D Dataset visualization – Beam height [mm] (abscissa) vs Column web thickness [mm] (ordinate)

The dataset is represented in Figures 5.8 and 5.11 respectively, by means of the pairs of features beam height (abscissa) and column web thickness (ordinates) and beam flange thickness (abscissa) and column web thickness (ordinates).

In the first 2D plot, the transition zone between the conditioning components presents itself less strict, with a superposition of the two components, column web compression and column flange in bending as well as with the web shear component.
From the connection's structural behaviour point-of-view, as it would be expected, smaller column web thicknesses govern the column web compression component. The obtained correlation between beam height and beam flange thickness (0.74), suggests that for a considerable part of cases, the increase in the beam's height leads to the corresponding increase of the beam's flange thickness and thus to it resistance. Thus, considering a constant column web thickness, the successive increase of the beam's height leads to a change in the conditioning component, from the column flange in bending component (for smaller beam flange thicknesses) to the column web compression (for larger beam flange thicknesses).

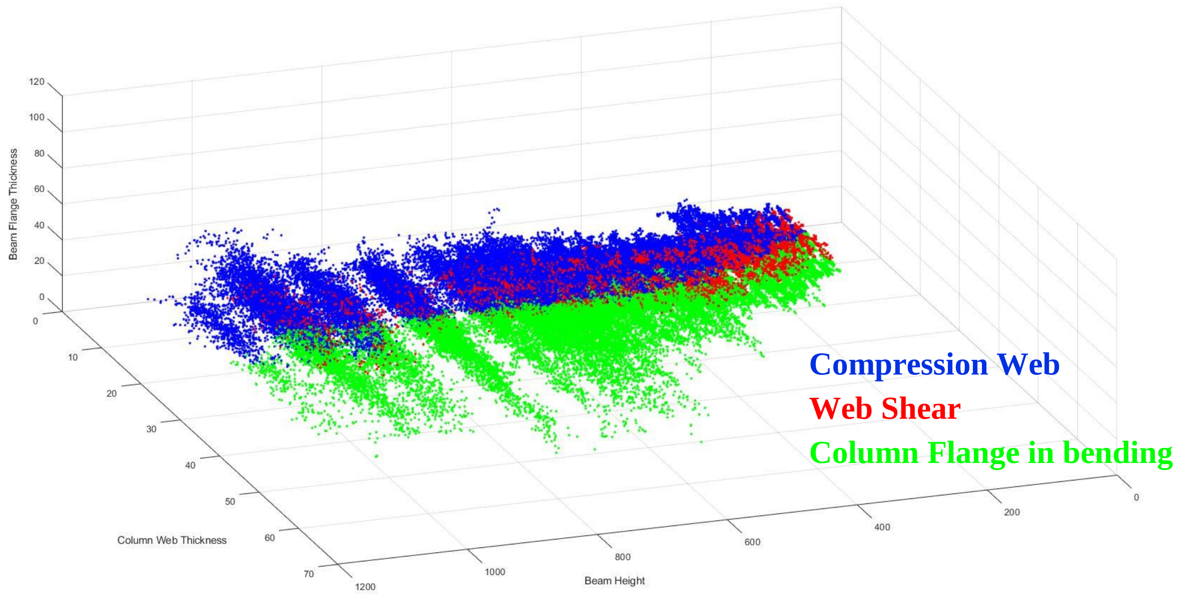


Figure 5.9 – 3D Dataset visualization – Beam height [mm] (abscissa) vs Column web thickness [mm] (ordinate) vs Beam flange thickness [mm] (applicate)

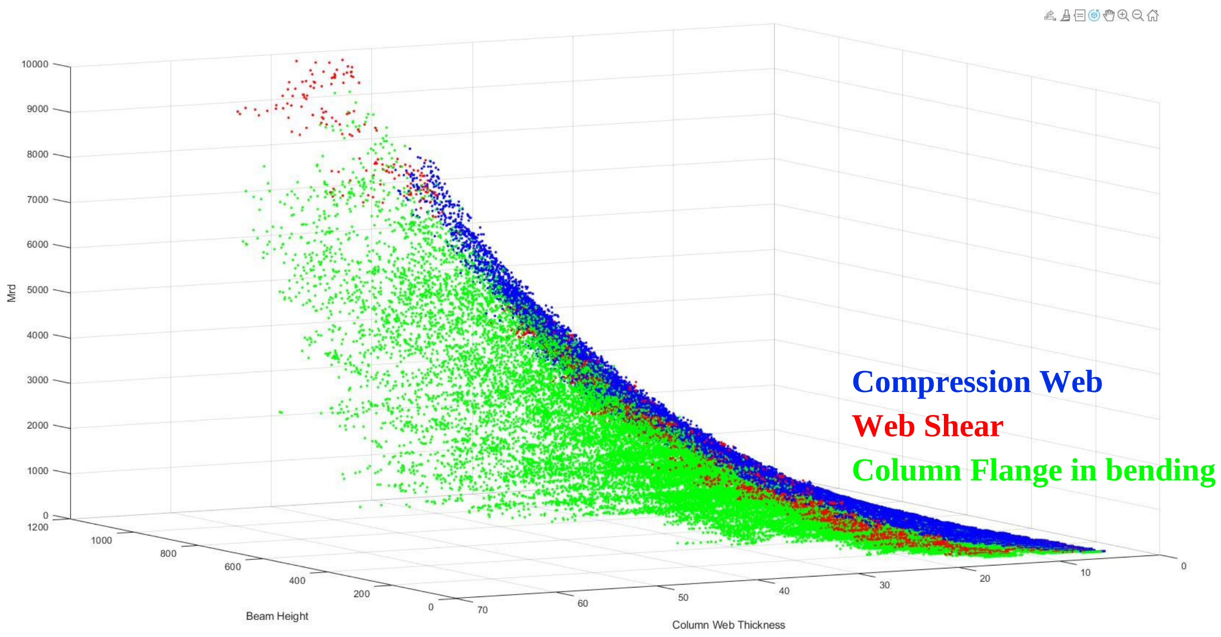


Figure 5.10 – 2D Dataset visualization – Beam height [mm] (abscissa) vs Column web thickness [mm] (ordinate) vs Mrd [kN.m] (applicate)

In the second plot, the one in Figure 5.11, the transition zone between the column web compression and beam flange compression appears more clear, less blurred, suggesting that the features adopted for this representation of the dataset, and in particular the beam flange thickness, are more adequate, and are thus a more proper indirect measure of the beam's resistance.

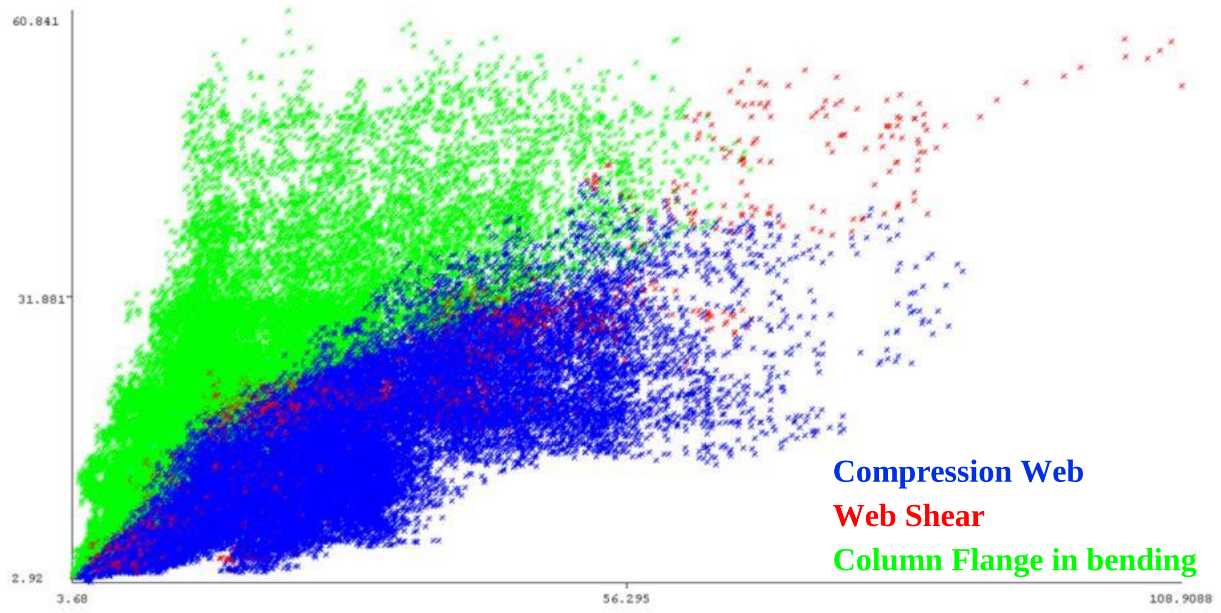


Figure 5.11 – 2D Dataset visualization – Beam flange thickness [mm] (abscissa) vs Column web thickness [mm] (ordinate)

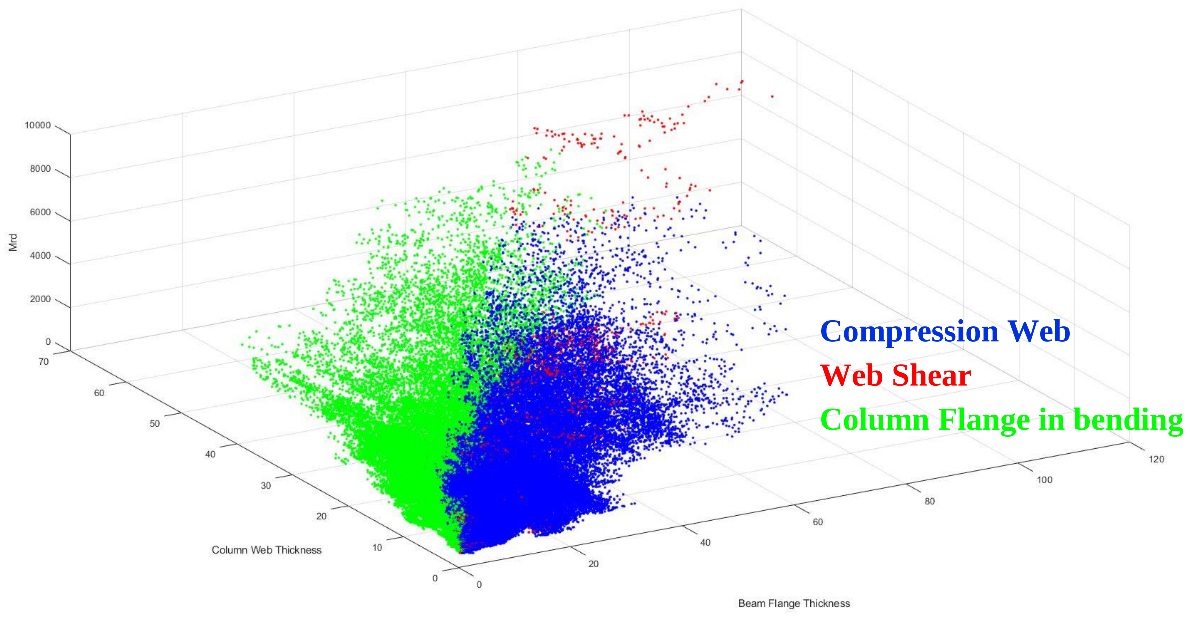


Figure 5.12 – 2D Dataset visualization – Beam flange thickness [mm] (abscissa) vs Column web thickness [mm] (ordinate) vs Mrd [kN.m] (applicate)

Once more, the consideration of an increased number of dimensions, between the 2D plots of Figures 5.8 and 5.11 and their respective 3D counterparts in Figures 5.9, 5.10 and 5.12 appears to return clearer separation boundaries between the different classes considering the same base features in the abscissa and ordinate axis.

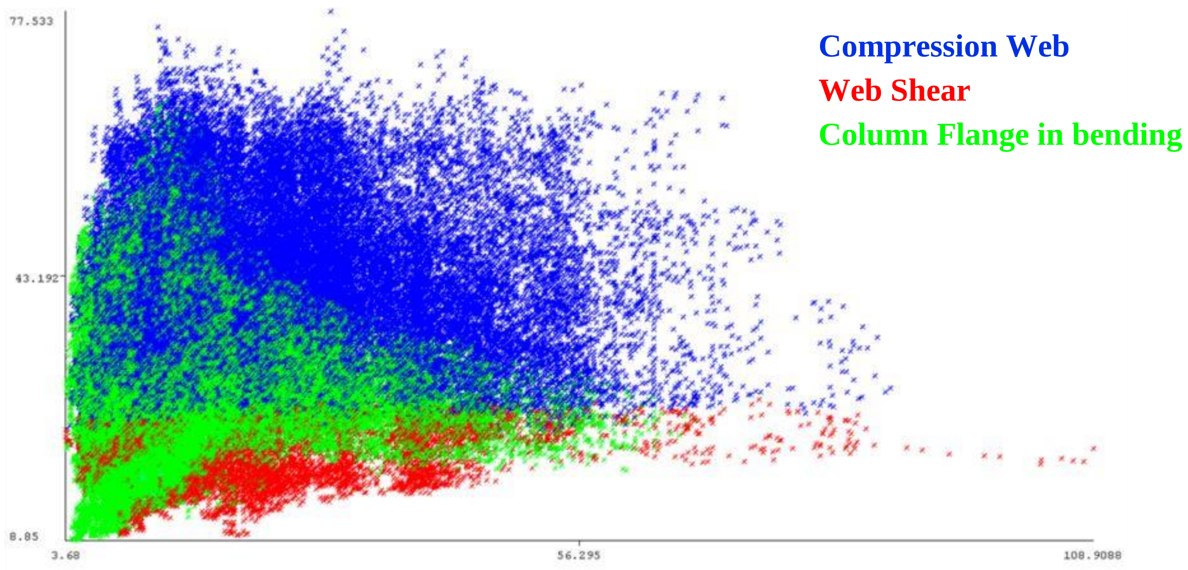


Figure 5.13 – Dataset visualization – Beam flange thickness [mm] (abscissa) vs Column height/Column web thickness ratio (ordinate)

In an attempt to introduce a third feature in a 2D plot and a fourth feature in a 3D representation of the dataset that would allow for a better representation of the web shear component, a "temporary" input was considered. Considering the influence of both the column height and the column web thickness in the resistance of this particular component, the ratio column height/column web thickness was considered.

As it can be seen from Figure 5.13 and 5.14, despite some superposition between data points of different classes, the obtained transition zone between web shear and two remaining components is now much clear.

It is noticeable that for lower column height to column web thickness ratios, the design resistance of the connection is governed by web shear, except for smaller beam heights.

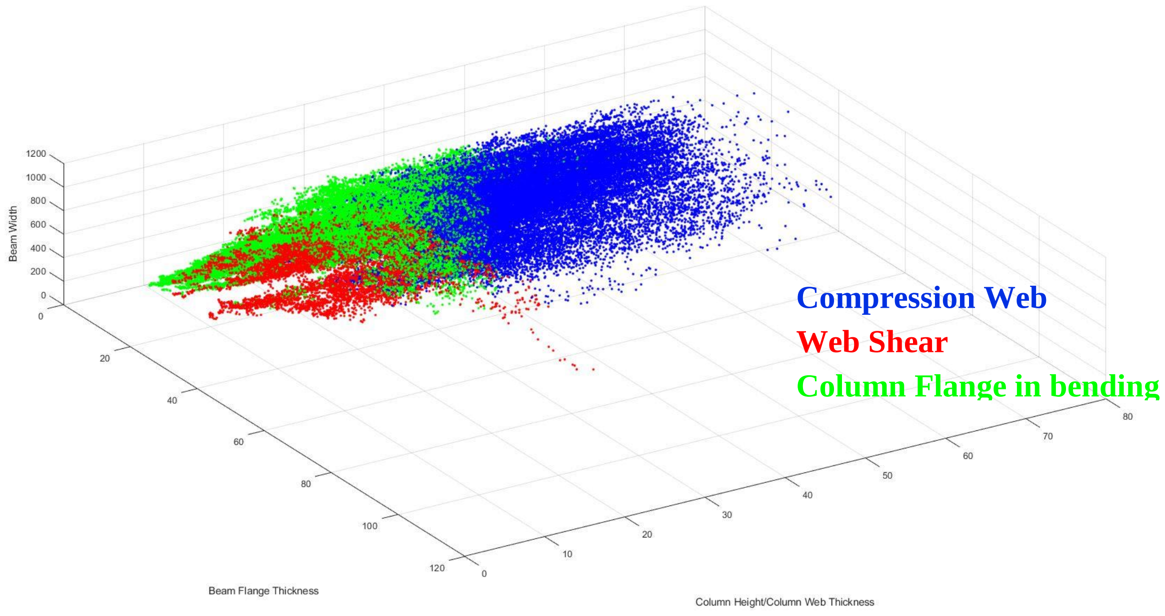


Figure 5.14 – Dataset visualization – Beam flange thickness [mm] (abscissa) vs Column height/Column web thickness ratio (ordinate) vs Beam width [mm] (applicate)

5.2 Component Classification

In order to address the classification task, in which the conditioning component is to be predicted, four different learning algorithms have been selected, namely, Decision Trees, k-Nearest Neighbors, Support Vector Machines and Neural Networks.

Before applying any of the above-mentioned learning algorithms, the dataset was split into a training set and a testing set in a 70/30 proportion.
Once these datasets, comprised by all 8 features were obtained, it was possible to obtain new pairs of training sets and testing sets by successively eliminating specific features in order to evaluate their influence in the learning algorithms behaviour.

5.2.1 Decision Trees

The use of Decision Trees for the classification task was made by means of the sklearn library function DecisionTreeClassifier(), allowing not only a graphic representation of the decision path but also the obtainment of its full depth, the total number of leaves as well as the obtained precision, recall and F1-score for the three classes as well as the overall accuracy of the model.

Table 5.1 presents the confusion matrix obtained considering the dataset with all 8 previously mentioned features. The corresponding Decision Tree is represented graphically in Figures 5.15 and 5.16, the first showing the first two decision nodes and the latter the first four nodes.

Component		Prediction Value		
		Column Flange in bending	Compression Web	Web Shear
Actual Value	Column Flange in bending	10203	397	90
	Compression Web	351	16509	110
	Web Shear	93	91	2112

Table 5.1 – Decision Tree Classifier - Confusion matrix – Test set with all 8 features

From the analysis of the Confusion Matrix, it is possible to conclude that from the 10690 (10203+397+90) examples that correspond effectively to the Column Flange in bending class, 10203 are correctly classified. Following further down along the confusion matrix diagonal, on verifies that from the 16970 (351+16509+110) Compression Web datapoints, 16509 are correctly predicted as such and 2112 are correctly classified as Web Shear amongst a total of 2296.

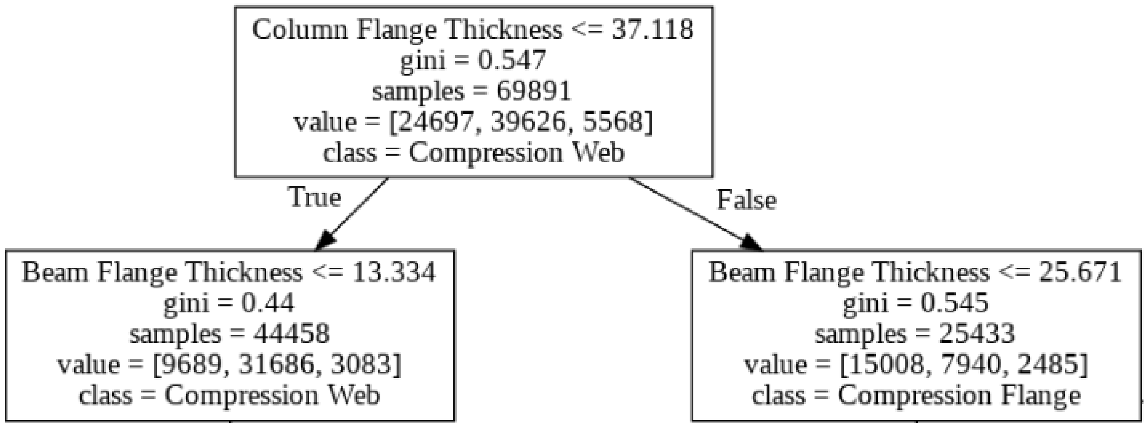

Column Flange Thickness <= 37.118
gini = 0.547
samples = 69891
value = [24697, 39626, 5568]
class = Compression Web
True
False
Beam Flange Thickness <= 13.334
gini = 0.44
samples = 44458
value = [9689, 31686, 3083]
class = Compression Web
Beam Flange Thickness <= 25.671
gini = 0.545
samples = 25433
value = [15008, 7940, 2485]
class = Compression Flange

Figure 5.15 – Decision tree Classifier (first two nodes) – Test set with all 8 features

A new confusion matrix, also obtained following the use of Decision Tree, is presented in Table 5.2. This matrix, contrarily to what was considered for the matrix presented in Table 5.1, does not consider all the features, instead 5 features were removed, namely, beam height h_b and web thickness tw_b, column height h_c and width b_c as well as the column flange thickness tf_c.

Component		Prediction Value		
		Column Flange in bending	Compression Web	Web Shear
Actual Value	Column Flange in bending	9861	511	318
	Compression Web	466	15367	1137
	Web Shear	345	1082	869

Table 5.2 – Decision Tree Classifier - Confusion matrix – Test set without beam height h_b and web thickness tw_b, column height h_c, width b_c and flange thickness tf_c

Analyzing the confusion matrix in Table 5.2 and comparing it to the one from Table 5.1, it is noticeable the general decrease in correctly classified examples, in particular the ones related to the Web Shear class, the one with the smaller amount of datapoints.

Column Flange in bending	Precision	Recall	F1-score	Support
With all 8 features	0.96	0.96	0.96	10690
Without h_b, tw_b, b_c, tf_c	0.96	0.96	0.96	
Without h_b, tw_b, h_c, b_c, tf_c	0.92	0.92	0.92	

Table 5.3 – Decision Tree Classifier - Comparison of Column Flange in bending Precision, Recall and F1-score

Compression Web	Precision	Recall	F1-score	Support
With all 8 features	0.97	0.97	0.97	16970
Without h_b, tw_b, b_c, tf_c	0.97	0.97	0.97	
Without h_b, tw_b, h_c, b_c, tf_c	0.91	0.91	0.91	

Table 5.4 – Decision Tree Classifier - Comparison of Column Flange in bending Precision, Recall and F1-score

Web Shear	Precision	Recall	F1-score	Support
With all 8 features	0.92	0.93	0.92	2296
Without h_b, tw_b, b_c, tf_c	0.93	0.93	0.93	
Without h_b, tw_b, h_c, b_c, tf_c	0.38	0.38	0.38	

Table 5.5 – Decision Tree Classifier - Comparison of Column Flange in bending Precision, Recall and F1-score

Throughout tables 5.3 to 5.5, three different evaluation metrics, related specifically to each one of the existing classes are presented, namely Precision, Recall and F1-score. The variation of these metrics with the variation of the number of considered features is also presented in each table. It should be mentioned the good overall results obtained for the different classes while considering the 8 features, as well as the steep decrease of the different metrics for the Web Shear component after the removal of the column height feature h_c.

	Accuracy	Support
With all 8 features	0.96	29956
Without h_b, tw_b, b_c, tf_c	0.96	
Without h_b, tw_b, h_c, b_c, tf_c	0.87	

Table 5.6 – Decision Tree Classifier - Comparison of Accuracy

5.2.2 k-Nearest Neighbors

The implementation of the k-Nearest Neighbors (kNN) algorithm was achieved through the KNeighborsClassifier() function. This function presents as its main parameter the number of k neighbour examples to be considered in the voting process for the classification of a previously unseen datapoint.

In order to verify the suitability of the application of the kNN algorithm to the design of steel connections and in particular to the classification of the conditioning component, a sensitivity analysis was developed by essentially varying the number of k neighbour examples.

Tables 5.7, 5.8, 5.9 and 5.10 present different confusion matrixes obtained respectively for k=1,3,5 and 7.

Component		Prediction Value		
		Column Flange in bending	Compression Web	Web Shear
Actual Value	Column Flange in bending	10110	523	57
	Compression Web	449	16385	136
	Web Shear	71	120	2105

Table 5.7 – kNN Classifier - Confusion matrix – Test set with all 8 features and k=1

Component		Prediction Value		
		Column Flange in bending	Compression Web	Web Shear
Actual Value	Column Flange in bending	10011	614	65
	Compression Web	484	16346	140
	Web Shear	92	120	2084

Table 5.8 – kNN Classifier - Confusion matrix – Test set with all 8 features and k=3

Component		Prediction Value		
		Column Flange in bending	Compression Web	Web Shear
Actual Value	Column Flange in bending	9892	738	60
	Compression Web	529	16313	128
	Web Shear	107	149	2040

Table 5.9 – kNN Classifier - Confusion matrix – Test set with all 8 features and k=5

Component		Prediction Value		
		Column Flange in bending	Compression Web	Web Shear
Actual Value	Column Flange in bending	9735	882	73
	Compression Web	567	16270	133
	Web Shear	127	191	1978

Table 5.10 – kNN Classifier - Confusion matrix – Test set with all 8 features and k=7

From the analysis of these matrixes together with the Precision, Recall and F1-score metrics presented in table 5.11 throughout 5.13, it is noticeable that good results can be achieved by considering the single closest k=1 example, while an increase in the number of considered neighbours leads to a general decrease in the quality of the results, which is also supported by the reduction in the overall Accuracy presented in Table 5.14.

Column Flange in bending	Precision	Recall	F1-score	Support
k=1	0.95	0.95	0.95	10690
k=3	0.95	0.94	0.94	
k=5	0.94	0.93	0.93	
k=7	0.93	0.91	0.92	

Table 5.11 – kNN Classifier - Comparison of Column Flange in bending Precision, Recall and F1-score

Compression Web	Precision	Recall	F1-score	Support
k=1	0.96	0.97	0.96	16970
k=3	0.96	0.96	0.96	
k=5	0.95	0.96	0.95	
k=7	0.94	0.96	0.95	

Table 5.12 – kNN Classifier - Comparison of Column Flange in bending Precision, Recall and F1-score

Web Shear	Precision	Recall	F1-score	Support
k=1	0.92	0.92	0.92	2296
k=3	0.91	0.91	0.91	
k=5	0.92	0.89	0.90	
k=7	0.91	0.86	0.88	

Table 5.13 – kNN Classifier - Comparison of Web Shear Precision, Recall and F1-score

	Accuracy	Support
k=1	0.95	29956
k=3	0.95	
k=5	0.94	
k=7	0.93	

Table 5.14 – kNN Classifier - Comparison of Accuracy

5.2.3 Support Vector Machines – Linear Kernel

The application of learning algorithms known as Support Vector Machine (SVM) to the classification problem being undertaken was initially made using the linear kernel. In order to do so, the SVC() function of the sklearn library was considered.
Previously to the use of the SVC() function it was necessary to engage in a standardization process of the different considered features, a process made possible by means of the StandardScaler() function applied to the trainingset. The parameters of the standardization of the trainingset were then later used for the standardization of the examples that compose the testset. Once both dataset are standardized, it was possible to develop an analysis that would allow a deeper understanding of the suitability of this learning algorithm to the proposed problem, namely by varying the c parameter, defined as the regularization parameter. A decrease of c leads to a larger regularization and hence to a larger margin at the cost of some accuracy.

Component		Prediction Value		
		Column Flange in bending	Compression Web	Web Shear
Actual Value	Column Flange in bending	10121	471	98
	Compression Web	369	16502	99
	Web Shear	146	78	2072

Table 5.15 – SVM Linear Kernel Classifier - Confusion matrix –Test set with all 8 features and c=1

Table 5.15 presents the confusion matrix for the dataset containing all 8 features and assuming the unit value for the regularization parameter c=1.

Column Flange in bending	Precision	Recall	F1-score	Support
c=0.001	0.92	0.83	0.88	10690
c=0.01	0.95	0.93	0.94	
c=0.1	0.95	0.94	0.95	
c=1	0.95	0.95	0.95	
c=10	0.95	0.95	0.95	
c=100	0.95	0.95	0.95	
c=1000	0.95	0.95	0.95	

Table 5.16 – SVM Linear Kernel Classifier - Comparison of Column Flange in bending Precision, Recall and F1-score

Compression Web	Precision	Recall	F1-score	Support
c=0.001	0.86	0.98	0.92	16970
c=0.01	0.93	0.98	0.95	
c=0.1	0.96	0.98	0.97	
c=1	0.97	0.97	0.97	
c=10	0.97	0.97	0.97	
c=100	0.97	0.97	0.97	
c=1000	0.97	0.97	0.97	

Table 5.17 – SVM Linear Kernel Classifier - Comparison of Compression Web Precision, Recall and F1-score

Web Shear	Precision	Recall	F1-score	Support
c=0.001	0.88	0.35	0.50	2296
c=0.01	0.93	0.66	0.77	
c=0.1	0.92	0.87	0.89	
c=1	0.91	0.90	0.91	
c=10	0.91	0.91	0.91	
c=100	0.91	0.91	0.91	
c=1000	0.91	0.91	0.91	

Table 5.18 – SVM Linear Kernel Classifier - Comparison of Web Shear Precision, Recall and F1-score

Tables 5.16 throughout 5.18 present the Precision, Recall and F1-score metrics for each observed class as well as its variation with the regularization parameter. It should be noticed the slight increase of the quality of the results along with the increase of the regularization parameter, at least until c approaches the unit value (c=1).

	Accuracy	Support
c=0.001	0.88	29956
c=0.01	0.94	
c=0.1	0.96	
c=1	0.96	
c=10	0.96	
c=100	0.96	
c=1000	0.96	

Table 5.19 – SVM Linear Kernel Classifier - Comparison of Accuracy

5.2.4 Support Vector Machines – Radial Basis Function Kernel

In addition to the analysis presented above considering a SVM with linear kernel, it was also possible, taking advantage of the same SVC() function, to develop an analysis considering however a Radial Basis Function (RBF). In addition to the regularization parameter mentioned previously for the case of the SVM with the linear kernel, the SVC() function assuming a Radial Basis Function allows also the definition of the additional parameter γ, which can be understood as the parameter that defines the influence of a single example, thus while large values of γ correspond to small regions of influence which may lead to overfitting, the consideration of small values for this parameter may result in a model that is not capable of representing the complexity of the training examples.

Component		Prediction Value		
		Column Flange in bending	Compression Web	Web Shear
Actual Value	Column Flange in bending	10450	210	30
	Compression Web	156	16732	82
	Web Shear	34	107	2155

Table 5.20 – SVM RBF Kernel Classifier - Confusion matrix –Test set with all 8 features and c=10 and γ=10

Table 5.20 presents the confusion matrix obtained considering the standardized dataset with all 8 features and assuming a value of 10 for both c and γ parameters.
From the analysis of the above confusion matrix, it is possible to observe that the use of SVM learning algorithm considering a RBF leads to good overall results, especially for the case of the more represented classes such as in the case of Column Flange in bending and Compression Web.

Column Flange in bending		Precision	Recall	F1-score	Support
c=0.1	γ=0.1	0.96	0.95	0.96	10690
	γ=1	0.97	0.96	0.96	
	γ=10	0.97	0.93	0.95	
c=1	γ=0.1	0.98	0.97	0.97	
	γ=1	0.98	0.98	0.98	
	γ=10	0.98	0.97	0.98	
c=10	γ=0.1	0.98	0.98	0.98	
	γ=1	0.99	0.98	0.99	
	γ=10	0.98	0.98	0.98	

Table 5.21 – SVM RBF Kernel Classifier - Comparison of Column Flange in bending Precision, Recall and F1-score

Compression Web		Precision	Recall	F1-score	Support
c=0.1	γ=0.1	0.95	0.98	0.97	16970
	γ=1	0.96	0.98	0.97	
	γ=10	0.93	0.99	0.96	
c=1	γ=0.1	0.97	0.99	.98	
	γ=1	0.98	0.99	0.98	
	γ=10	0.98	0.98	0.98	
c=10	γ=0.1	0.98	0.99	0.99	
	γ=1	0.99	0.99	0.99	
	γ=10	0.98	0.99	0.98	

Table 5.22 – SVM RBF Kernel Classifier - Comparison of Compression Web Precision, Recall and F1-score

Web Shear		Precision	Recall	F1-score	Support
c=0.1	γ=0.1	0.94	0.76	0.84	2296
	γ=1	0.93	0.83	0.88	
	γ=10	0.96	0.65	0.78	
c=1	γ=0.1	0.94	0.88	0.91	
	γ=1	0.94	0.91	0.92	
	γ=10	0.94	0.92	0.93	
c=10	γ=0.1	0.93	0.93	0.93	
	γ=1	0.95	0.95	0.95	
	γ=10	0.95	0.94	0.94	

Table 5.23 – SVM RBF Kernel Classifier - Comparison of Web Shear Precision, Recall and F1-score

Tables 5.21 and 5.22 show the results for the different considered metrics (Precision, Recall and F1-score) for the analysis of specific classes, in this case the Column Flange in bending and Compression Web classes respectively and their variation with both c and γ parameters.
For these two classes the obtained results are very good, in particular when considering c=10 and γ=1. These results are also corroborated by the overall accuracy displayed on Table 5.24.

		Accuracy	Support
c=0.1	γ=0.1	0.95	29956
	γ=1	0.96	
	γ=10	0.94	
c=1	γ=0.1	0.97	
	γ=1	0.98	
	γ=10	0.97	
c=10	γ=0.1	0.98	
	γ=1	0.98	
	γ=10	0.98	

Table 5.24 – SVM RBF Kernel Classifier - Comparison of Accuracy

5.2.5 Artificial Neural Networks

The development of a model using Artificial Neural Networks (ANN) was made possible by means of the MLPClassifier() function. This function presents as its main inputs the number of hidden layers which will compose the network as well as the type of activation function being considered for this case the logistic sigmoid function.
The stochastic gradient descent solver was considered leading to the need for the definition of other parameters such as the learning rate and the momentum and thus enabling the analysis of their influence on the obtained results. The number of times an example is used was also defined as an input, a parameter known as number of epochs.

Hidden Layers	Neurons	Class	Precision
1	5	Column Flange in bending	0.97
		Compression Web	0.98
		Web Shear	0.90
	10	Column Flange in bending	0.98
		Compression Web	0.99
		Web Shear	0.90
	15	Column Flange in bending	0.98
		Compression Web	0.99
		Web Shear	0.90
2	5	Column Flange in bending	0.95
		Compression Web	0.99
		Web Shear	0.88
	10	Column Flange in bending	0.96
		Compression Web	0.99
		Web Shear	0.88
3	5	Column Flange in bending	0.97
		Compression Web	0.99
		Web Shear	0.88
	10	Column Flange in bending	0.97
		Compression Web	0.99
		Web Shear	0.86
4	5	Column Flange in bending	0.96
		Compression Web	0.99
		Web Shear	0.87
	10	Column Flange in bending	0.97
		Compression Web	0.99
		Web Shear	0.86
5	5	Column Flange in bending	0.96
		Compression Web	0.95
		Web Shear	0.56

Table 5.25 – ANN Classifier – Comparison of Networks

The application of ANN to solve the proposed problem begun with a sensitivity analysis regarding the number of hidden layers as well as the number of neurons in each one of these layers and considering a fix learning rate and momentum of 0.3 and 0.2 respectively. Table 5.25 shows that even considering a reduced number of hidden layers and neurons good results were achieved. However, an increased number of hidden layers led to a decrease in the quality of the results especially for the less represented classes, in this case the Web Shear class.

Component		Prediction Value		
		Column Flange in bending	Compression Web	Web Shear
Actual Value	Column Flange in bending	10479	159	52
	Compression Web	164	16667	139
	Web Shear	99	46	2151

Table 5.26 – ANN Classifier - Confusion matrix –Test set with all 8 features and neural network with 3 hidden layers of 10 neurons each, a learning rate of 0.5 and a momentum of 0.9

Table 5.26 presents the obtained confusion matrix for a network composed of 3 hidden layers each one with 10 neurons and a learning rate and momentum of 0.5 and 0.9 respectively.

Column Flange in bending		Precision	Recall	F1-score	Support
L rate=0.1	Momentum=0.1	0.97	0.98	0.98	10690
	Momentum=0.5	0.98	0.98	0.98	
	Momentum=0.9	0.98	0.98	0.98	
L rate=0.5	Momentum=0.1	0.97	0.99	0.98	
	Momentum=0.5	0.98	0.99	0.98	
	Momentum=0.9	0.98	0.98	0.98	
L rate=0.9	Momentum=0.1	0.97	0.98	0.98	
	Momentum=0.5	0.98	0.98	0.98	
	Momentum=0.9	0.98	0.97	0.98	

Table 5.27 – ANN Classifier - Comparison of Column Flange in bending Precision, Recall and F1-score

Compression Web		Precision	Recall	F1-score	Support
L rate=0.1	Momentum=0.1	0.99	0.98	0.98	16970
	Momentum=0.5	0.99	0.98	0.98	
	Momentum=0.9	0.99	0.99	0.99	
L rate=0.5	Momentum=0.1	0.99	0.98	0.99	
	Momentum=0.5	0.99	0.98	0.99	
	Momentum=0.9	0.99	0.98	0.98	
L rate=0.9	Momentum=0.1	0.99	0.98	0.98	
	Momentum=0.5	0.99	0.98	0.99	
	Momentum=0.9	0.98	0.98	0.98	

Table 5.28 – ANN Classifier - Comparison of Compression Web Precision, Recall and F1-score

Web Shear		Precision	Recall	F1-score	Support
L rate=0.1	Momentum=0.1	0.90	0.95	0.92	2296
	Momentum=0.5	0.90	0.95	0.92	
	Momentum=0.9	0.92	0.95	0.93	
L rate=0.5	Momentum=0.1	0.90	0.95	0.93	
	Momentum=0.5	0.91	0.95	0.93	
	Momentum=0.9	0.92	0.94	0.93	
L rate=0.9	Momentum=0.1	0.90	0.95	0.93	
	Momentum=0.5	0.91	0.94	0.93	
	Momentum=0.9	0.90	0.94	0.92	

Table 5.29 – ANN Classifier - Comparison of Web Shear Precision, Recall and F1-score

For the sensitivity analysis regarding the variation of both the learning rate and the momentum the network used to obtain the confusion matrix of Table 5.25 was considered.
Tables 5.27 throughout 5.29 present the obtained results for the different combinations, considering a learning rate and momentum which could take a value of 0.1, 0.5 or 0.9.
From the analysis of the above mentioned tables it is possible to verify that the ANN learning algorithm led to overall good results, including the less represented class, which is still the one with the worst metrics.

		Accuracy	Support
L rate=0.1	Momentum=0.1	0.98	29956
	Momentum=0.5	0.98	
	Momentum=0.9	0.98	
L rate=0.5	Momentum=0.1	0.98	
	Momentum=0.5	0.98	
	Momentum=0.9	0.98	
L rate=0.9	Momentum=0.1	0.98	
	Momentum=0.5	0.98	
	Momentum=0.9	0.98	

Table 5.30 – ANN Classifier - Comparison of Accuracy

5.3 Resistant Bending Moment Regression

The prediction of the resistant bending moment of the steel connections between the different beam and columns pairs comprising the testing set was made considering models built mainly by means of the same learning algorithms used for the prediction of the conditioning components, although with the necessary modifications due to the different nature of classification and regression problems.
The examples comprising the training and testing set are the same used for the classification task with the exception of their respective outputs, which for the regression task are comprised of real-valued numbers.

The quality of the different models was evaluated by means of the Mean Absolute Percentage Error (MAPE) and also through a graphic representation of the results, in the form of 2D plots, was obtained. In these plots, the abscissa axis represents the actual values of the resistant bending moment determined accordingly to EN1993-1-8, while the ordinate axis represents the predicted resistant bending moment for the respective beam-column pair. An additional straight line in the form $y = x$, representing the set of perfectly predicted value was also added to the different plots.

5.3.1 Linear Regression

The use of Linear Regression for the development of a model to solve the regression task was done without any preprocessing of the data and was made possible by means of the LinearRegression() function present in the sklearn library.
From the analysis of Figure 5.16, which represents the results obtained with this algorithm, it is noticeable the large dispersion of the results relatively to the red line representing the perfectly predicted results.
It should also be mentioned two large deviations of the results. The first related to beam-column pairs with large resistant bending moments, and the second, at the opposite end of the spectrum, for connections with small resistant bending moments, where the predicted examples comprise negative values, incompatible with any physical interpretation for the behaviour of these connections.
The graphical representation of these results is corroborated by a MAPE=253%.

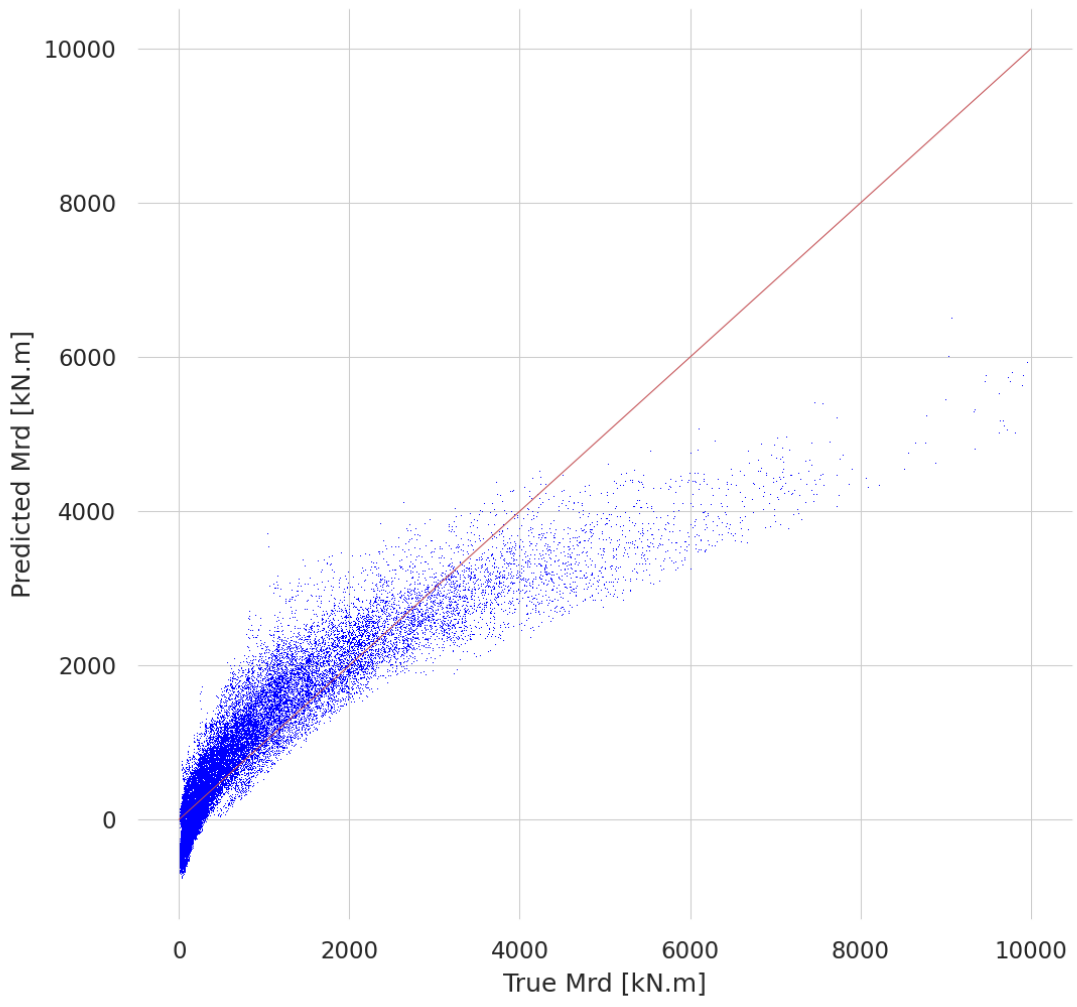


Figure 5.16 – Linear regression Plot - True Mrd [kN.m] (abscissas) vs Predicted Mrd [KN.m] (ordinates)

5.3.2 Decision Trees

The function DecisionTreeRegressor() applied to dataset without any preprocessing enabled the development of a regression model by means of Decision Trees.

The analysis of the results obtained with this learning algorithm was made considering the modification of the training and testsets by eliminating some of the features. This process allowed the evaluation of their influence on the behaviour of the algorithm.

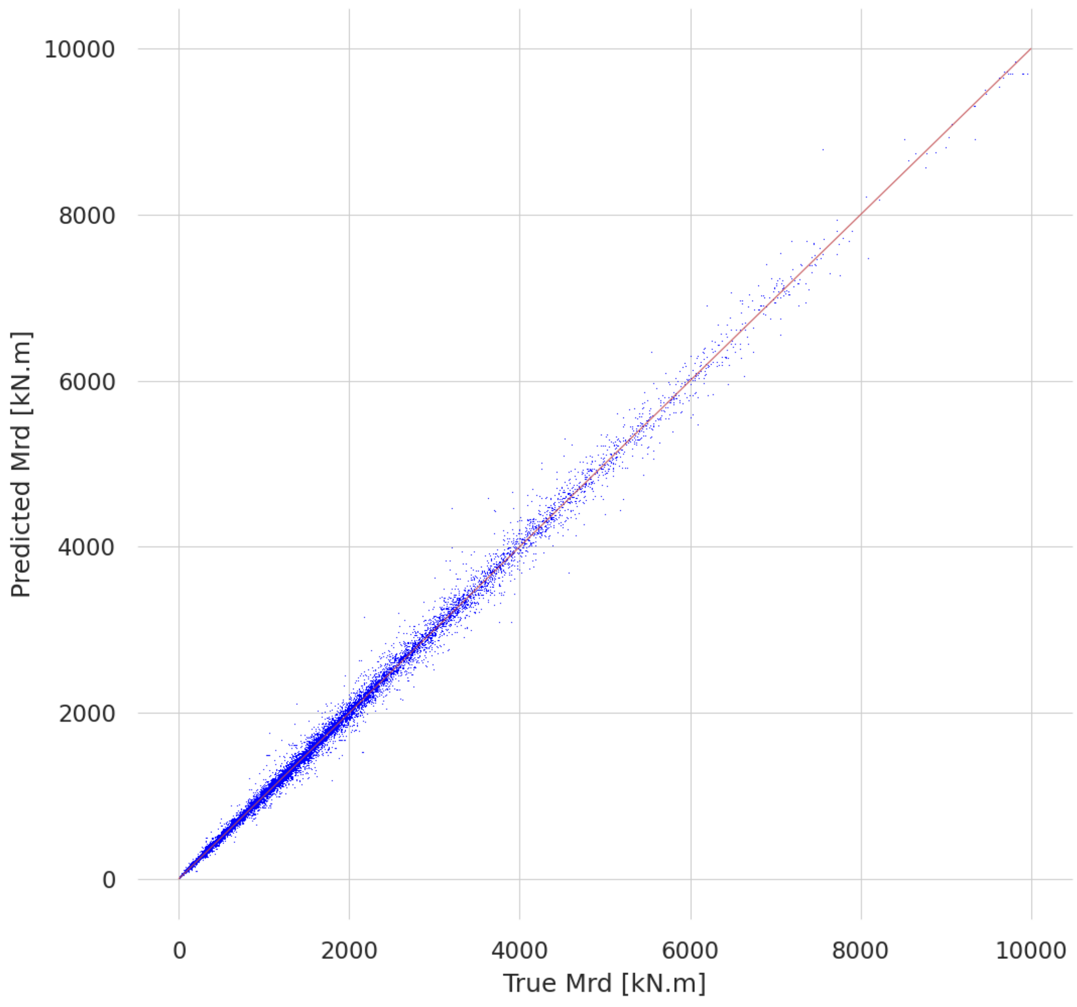


Figure 5.17 – Decision Tree Plot - True Mrd [kN.m] (abscissas) vs Predicted Mrd [KN.m] (ordinates) – All 8 features

The results represented on Figure 5.17, obtained considering all 8 features, are concentrated along the previously mentioned $y = x$ straight line, suggesting that this algorithm may be well suited for the regression problem, a conclusion also supported by the relatively small MAPE=2.8%, a value that is kept lower even without considering the beam web thickness tw_b (MAPE=2.8%) and both the beam web thickness tw_b and column flange thickness tf_c (MAPE=3.1%).

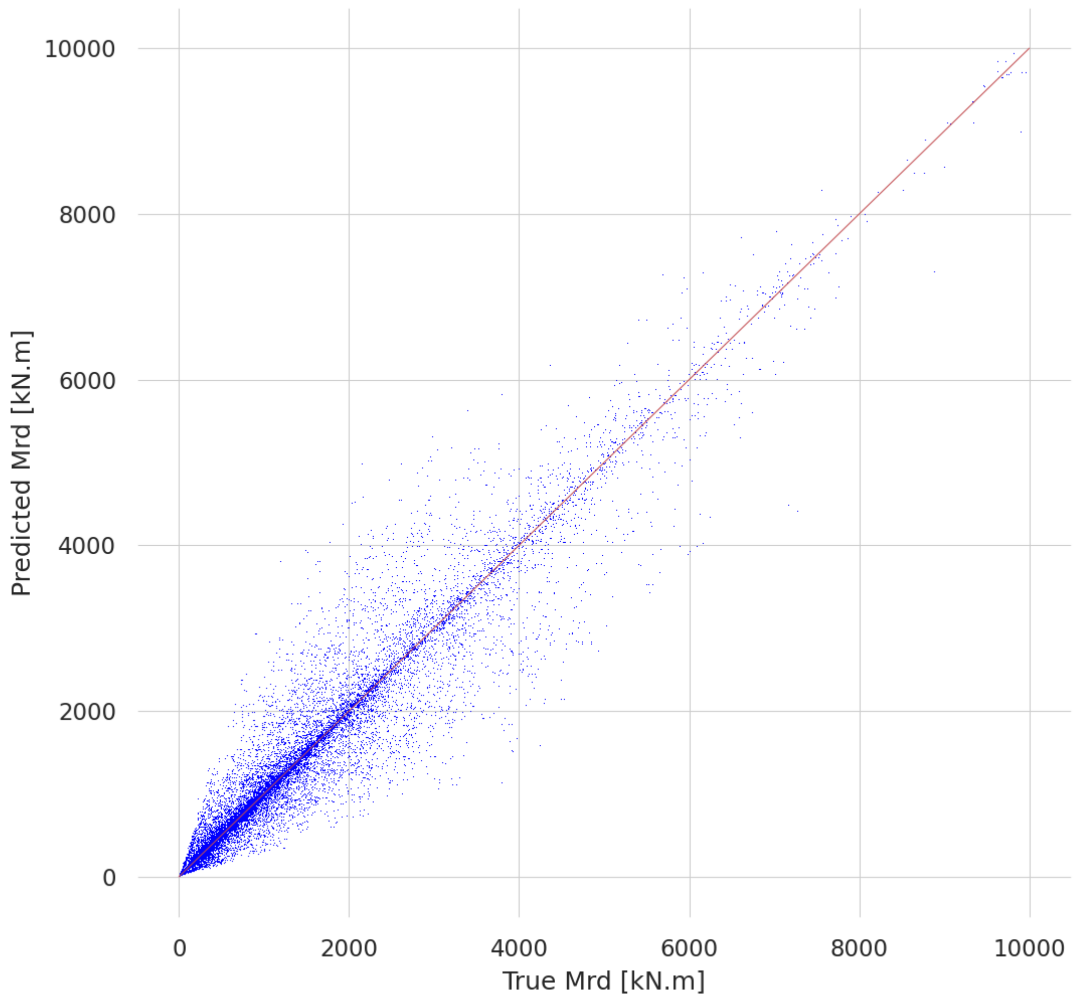


Figure 5.18 - Decision Tree Plot - True Mrd [kN.m] (abscissas) vs Predicted Mrd [KN.m] (ordinates) – Without beam height h_b and web thickness tw_b, column width b_c and flange thickness tf_c

The plot presented on Figure 5.18 shows the results obtained considering dataset without 4 of its original features, namely the beam web thickness tw_b, column flange thickness tf_c, beam height h_b and the column width b_c. In comparison to the plot on Figure 5.17, it is noticeable an increase in the dispersion of the results relatively to the $y = x$ straight line, also translated by the increase in the mean absolute percentage error of 13.9%.

5.3.3 k-Nearest Neighbors

The suitability of models created by means of the k-Nearest Neighbors (kNN) learning algorithm was evaluated through the application of the KNeighborsRegressor() function to the dataset. The analysis focused not only on the influence of the variation of the number of k neighbors to be considered but also by varying the number of considered features.

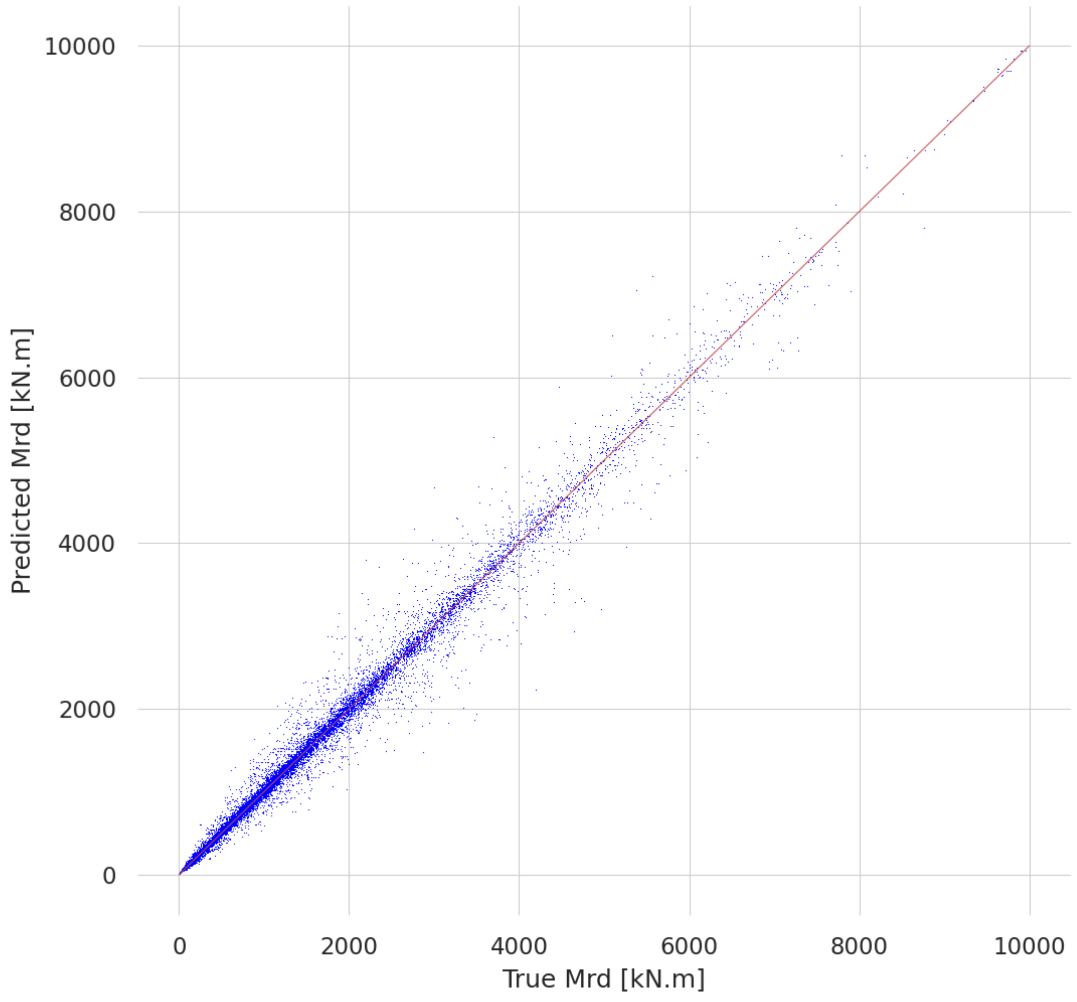


Figure 5.19 – k-Nearest Neighbors Plot - True Mrd [kN.m] (abscissas) vs Predicted Mrd [KN.m] (ordinates) – All 8 features and k=1

Figure 5.19 shows a plot with the obtained distribution of results considering a single neighbour k=1 and all 8 features, while the plot presented on Figure 5.20 shows the results

obtained with the same dataset but considering instead the sevens closest neighbors (k=7) to each new example.
The comparison between the two above mentioned plots, may not be enough to allow for a definite conclusion regarding which one of the created models best suits the problem. In this particular case the analysis of the mean absolute percentage error, respectively 4.6% and 8.3% leads to the conclusion that considering a single k=1 neighbor returns better results.

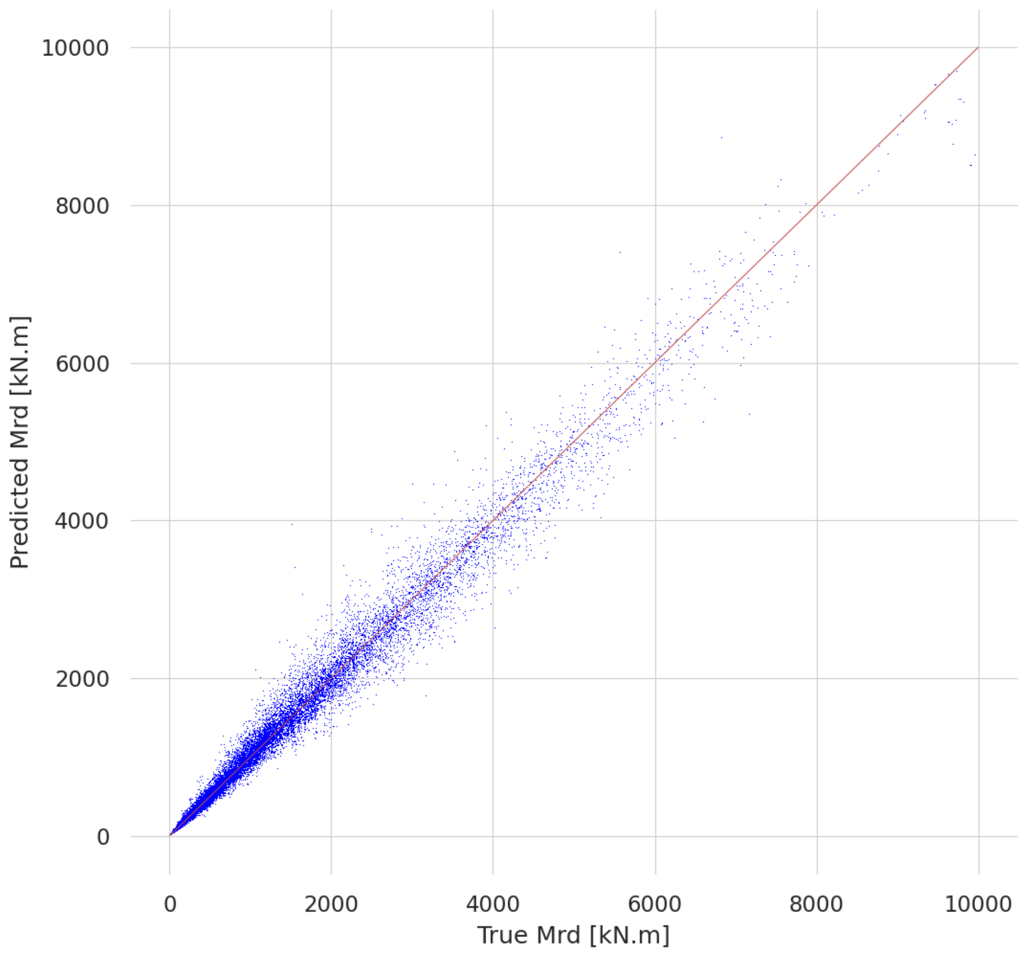


Figure 5.20 – k-Nearest Neighbors Plot - True Mrd [kN.m] (abscissas) vs Predicted Mrd [KN.m] (ordinates) – All 8 features and k=7

As for the case of the model created by means of Decision Trees, the model obtained through the kNN learning algorithm considering the removal of the features beam web thickness tw_b, column flange thickness tf_c, beam height h_b and the column width b_c leads to a considerable increase of the mean absolute percentage error value, MAPE=15.3%, as represented on Figure 5.21.

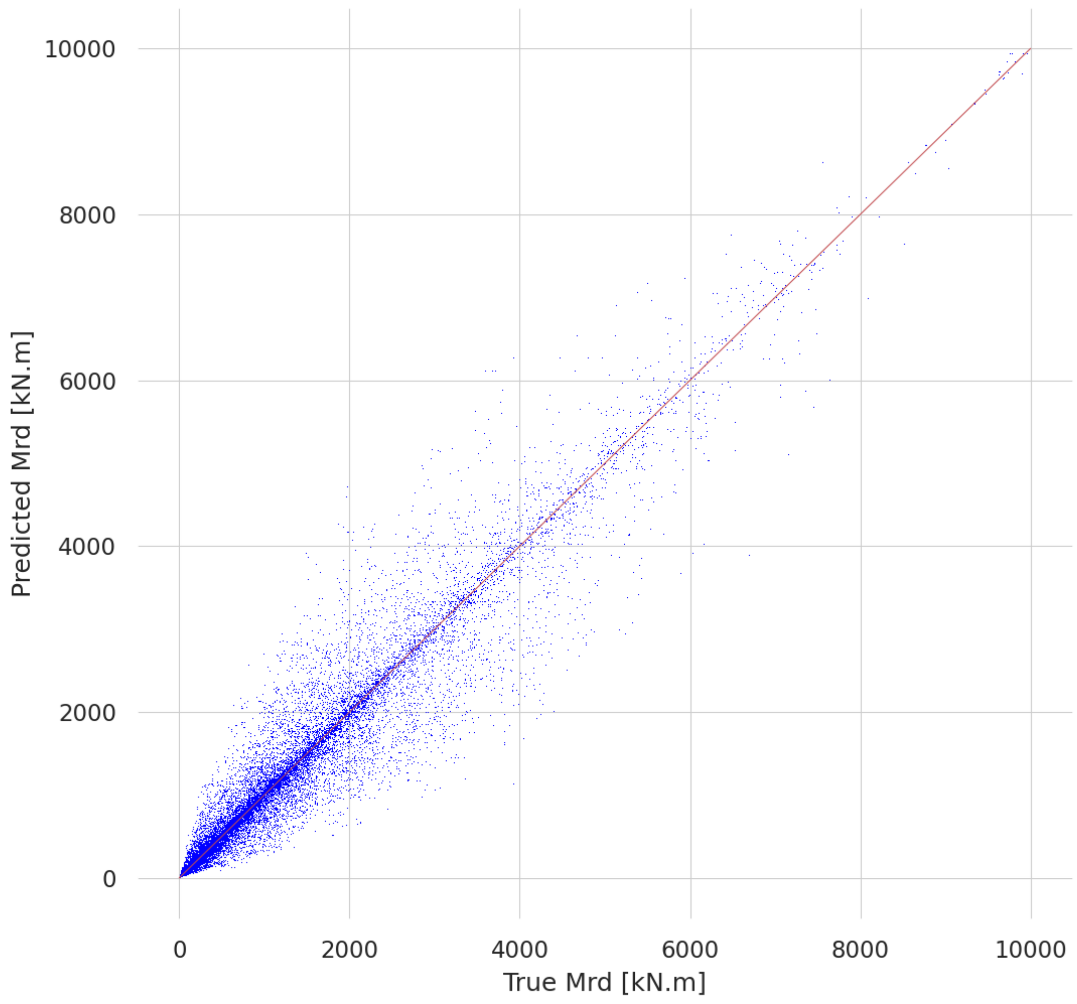


Figure 5.21 – k-Nearest Neighbors Plot - True Mrd [kN.m] (abscissas) vs Predicted Mrd [KN.m] (ordinates) – Without beam height h_b and web thickness tw_b, column width b_c and flange thickness tf_c and k=1

5.3.4 Support Vector Machines – Linear Kernel

The use of Support Vector Machines for the development of models to solve the regression problem was made possible by applying the SVR() function after the preprocessing of the datasets, namely through the standardization of the trainingset's features whose parameters where then used for the standardization of the testing set's features.
As for the case of the classification problem, the use in this case of the SVR() function led to need of using the regularization parameter c while considering a linear kernel. A sensitivity

analysis was developed in which the regularization parameter was successively considered to have the following values, 0.01, 0.1, 1, 10 and 100. Figures 5.22 and 5.23 show the graphical representation of the results obtained for the extreme values c=0.01 and c=100, respectively.

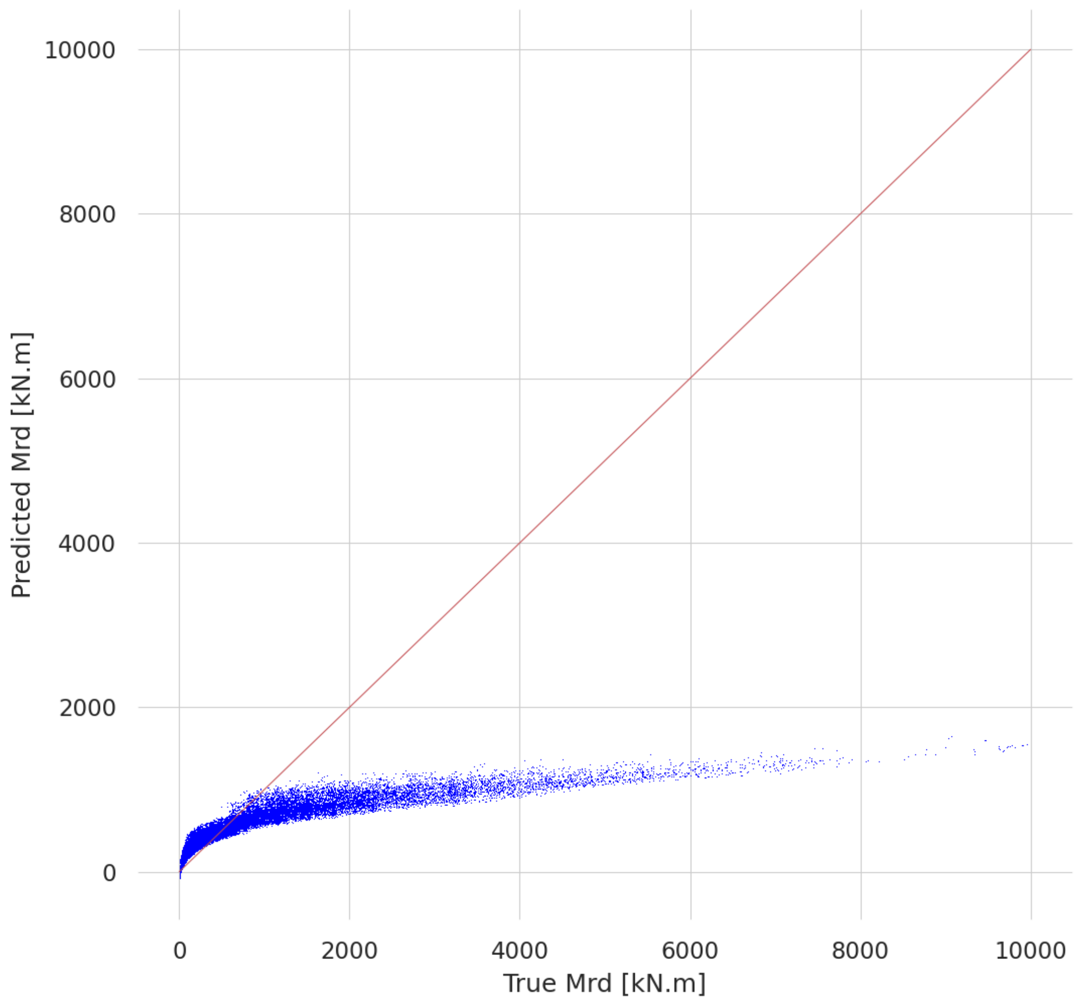


Figure 5.22 – SVM with Linear Kernel Plot - True Mrd [kN.m] (abscissas) vs Predicted Mrd [KN.m] (ordinates) – c=0.001

From the analysis of both figures, it should not be expected that Support Vector Machines with linear kernel may yield interesting results, a conclusion that is supported considering a mean absolute percentage error of MAPE=79.2% and MAPE=152.5%, respectively for c=0.01 and c=100.

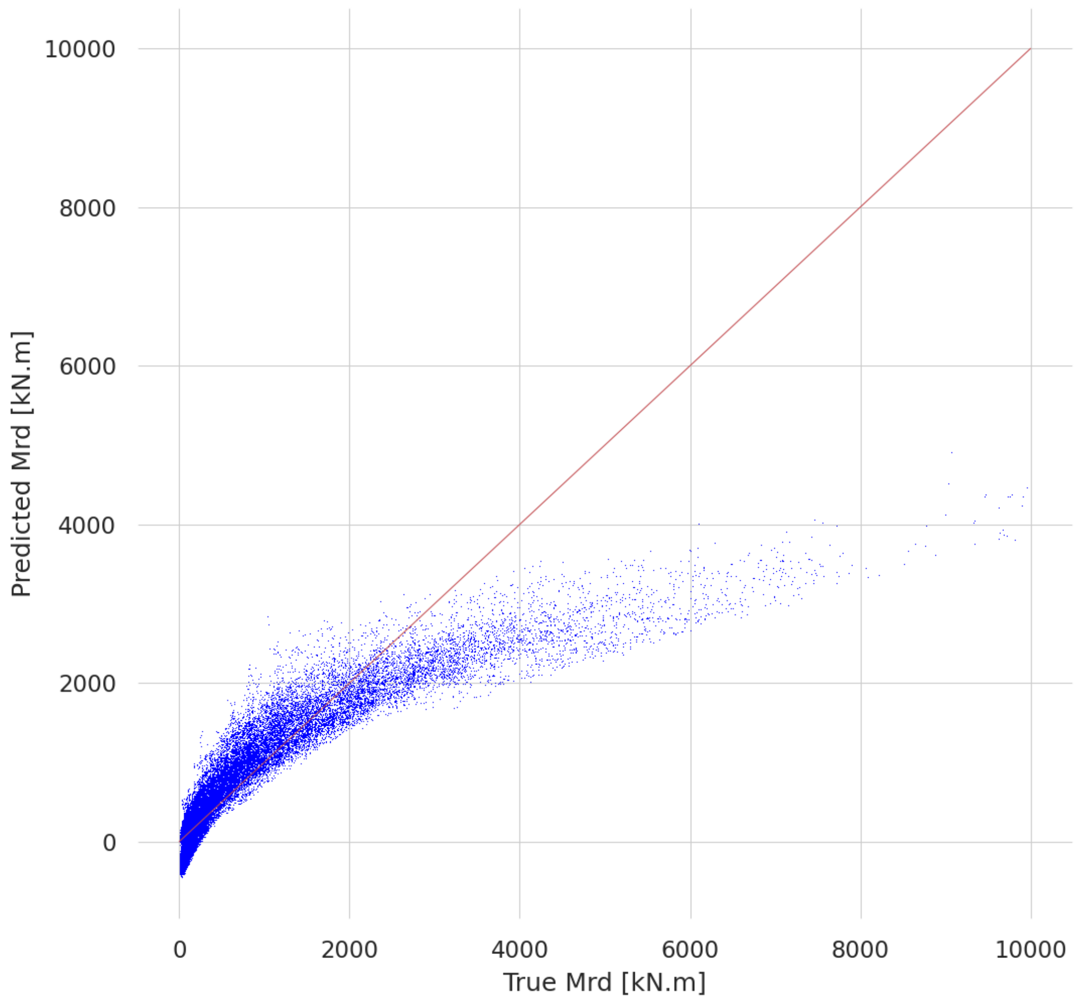


Figure 5.23 - SVM with Linear Kernel Plot - True Mrd [kN.m] (abscissas) vs Predicted Mrd [KN.m] (ordinates) – c=100

5.3.5 Support Vector Machines – Radial Basis Function Kernel

Once concluded the analysis assuming a linear kernel, the option for a Radial Basis Function was then considered.

Different analysis, in which all 8 features were taken into account, and that were a result of considering different combinations originated from c taking the values 0.1, 1, 10, 100 and 1000 and γ 0.1, 1 and 10 were developed.

Figure 5.24 shows the graphic representation of the obtained results for a model in which c=0.1 and γ=0.1, leading to results that considerably lack a proper accuracy, in particular for large values of resistant bending moments. These conclusion is also supported by a MAPE=77.1%.

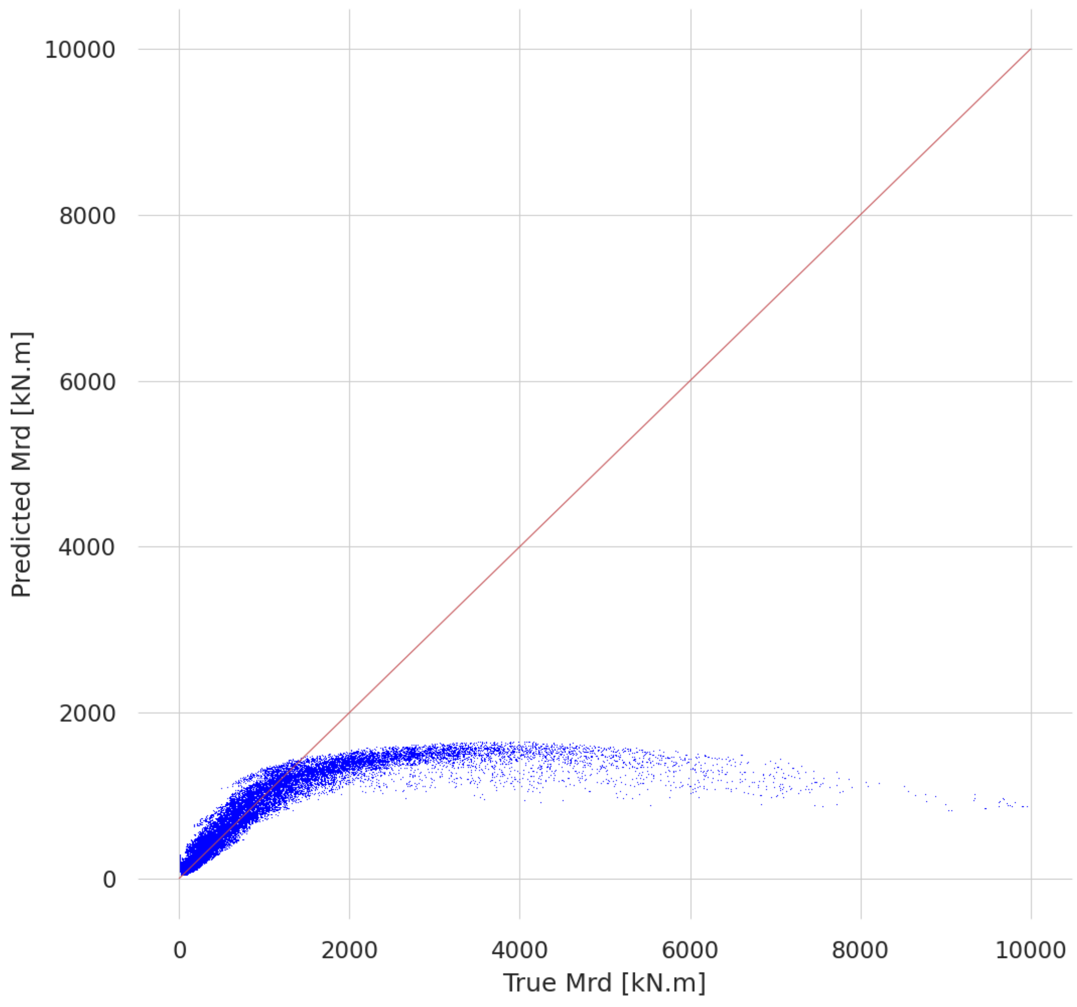


Figure 5.24 – SVM with Radial Basis Function Plot - True Mrd [kN.m] (abscissas) vs Predicted Mrd [KN.m] (ordinates) – c=0.1 and γ=0.1

The sensitivity analysis led however to interesting results when considering the pair of parameters c=1000 and γ=0.1. These results are graphically represented on image 5.25 and lead to a MAPE=2.6%.

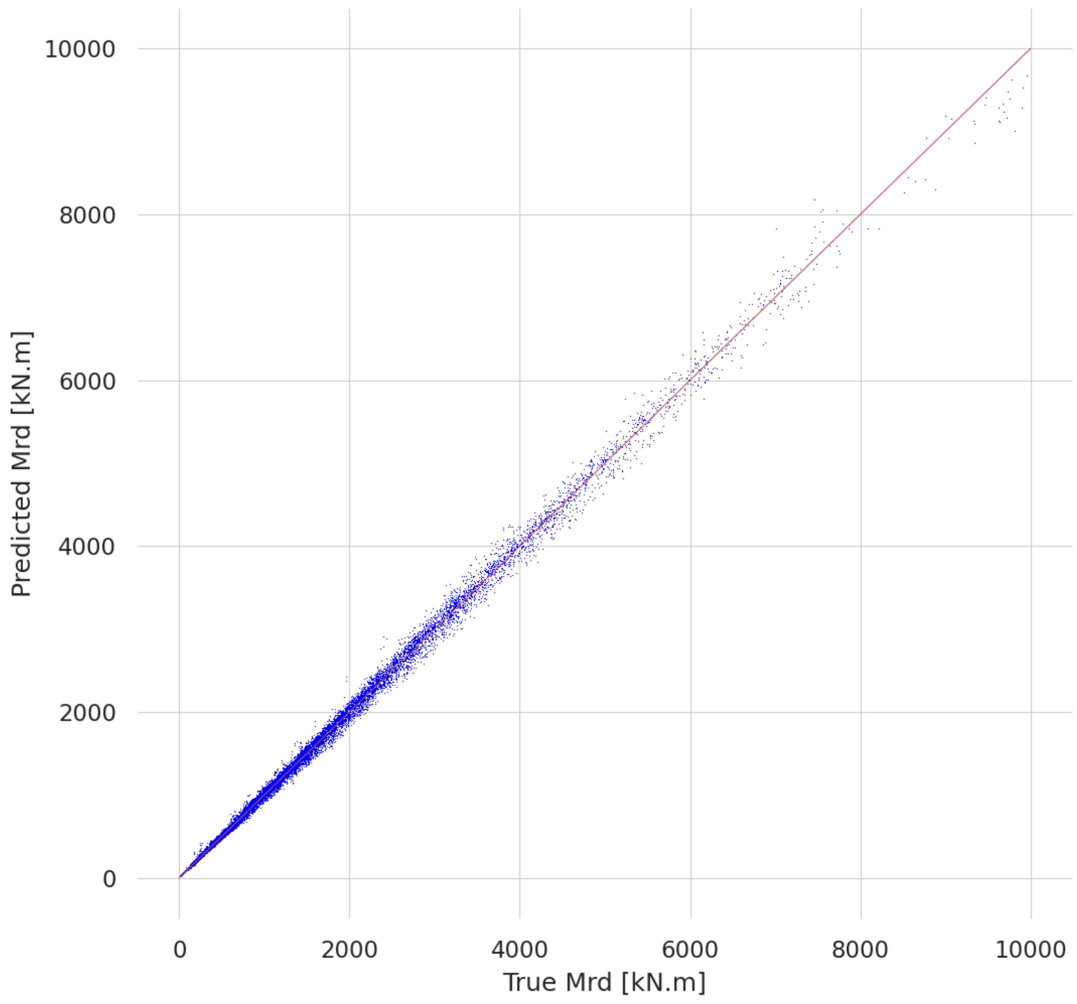


Figure 5.25 – SVM with Radial Basis Function Plot - True Mrd [kN.m] (abscissas) vs Predicted Mrd [KN.m] (ordinates) – c=1000 and γ=0.1

5.3.6 Artificial Neural Networks

The application of Artificial Neural Networks (ANN) to the regression problem was possible by means of the sklearn library function MLPRegressor().

The use of this function led to the need of a preprocessing procedures comprising not only the standardization of the features of each example but also their respective output.

In order to allow a correct representation of the resistant bending moment, an inversion process was need after the predicted results were obtained from the model, this task was accomplished by means of the inverse_transform() function.

As for the case of the classification task, the analysis by means of ANN begun with a sensitivity analysis by comparing the MAPE value of different network configurations, by varying the number of hidden layers and the number of neurons on each one of these layers while considering a fixed learning rate and momentum of 0.3 and 0.2 respectively.
The results are presented on Table 5.31 and show that the better results were obtained for a 2 hidden layer configuration with 10 neurons each and a 3 hidden layer configuration with 15 neurons each, resulting, respectively in mean absolute percentage errors of MAPE=21.6% and MAPE=17.8%.

Hidden Layers	Neurons	MAPE (%)
1	5	495 %
	10	26 %
	15	1398 %
2	5	211 %
	10	21.6 %
	15	26.7 %
3	5	455 %
	10	27 %
	15	17.8 %
4	5	189 %
	10	145 %
	15	23.5 %

Table 5.31 – ANN Regressor – Comparison of Networks

Taking into account the results presented on Table 5.31, the analysis focused on the two previously mentioned network configurations by varying the learning rate and momentum and considering that these parameters could take the values 0.005, 0.1, 0.5 or 0.9.

The results presented on the plot of Figure 5.26 correspond to the configuration the led to the best results, the 2 hidden layer network with 10 neurons and a learning rate of 0.005 and a momentum of 0.1, yielding a MAPE=14.7%

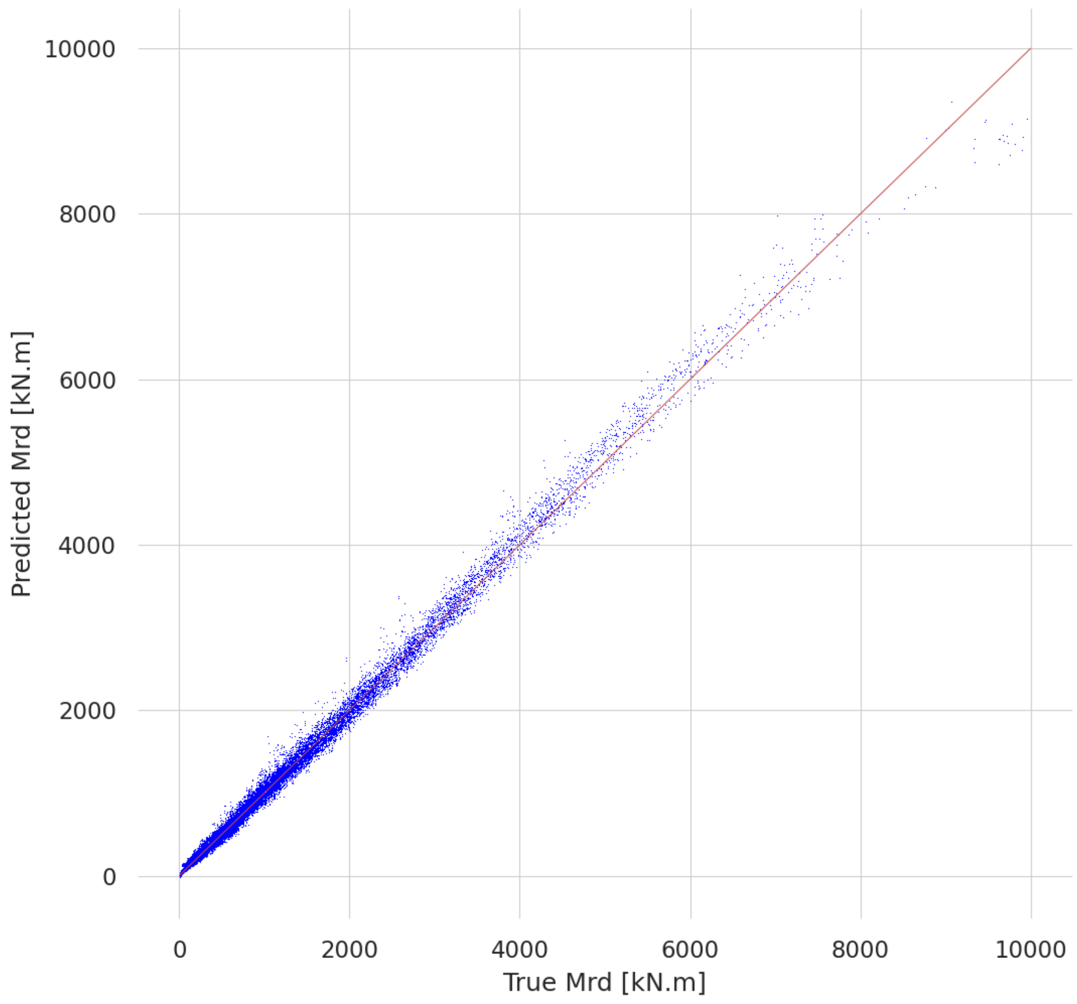


Figure 5.26 – ANN Plot - True Mrd [kN.m] (abscissas) vs Predicted Mrd [KN.m] (ordinates) – Testing set with all 8 features and neural network with 2 hidden layers of 10 neurons each, a learning rate of 0.005 and a momentum of 0.1

5.4 Summary

Once the sensitivity analysis of the different algorithms and their respective hyperparameters has been concluded for both the classification of the conditioning component and regression of the resistant bending moment, it is of interest to compare the quality of the obtained results. Table 5.32 presents a summary of the best results using each learning algorithm and for the classification and regression problems respectively.
For the classification problem, it is possible to obtain good predictions for algorithms considered, a conclusion that is supported by the overall high accuracy, in the range [0.96;0.98].

The regression problem presents a considerably higher dispersion of results, with low quality models created by means of Linear Regression and with Support Vector Machines with Linear Kernel and high quality models created through algorithms with different levels of complexity such as Decision Trees and Support Vector Machines with Radial Bases Function Kernel., these latter algorithms enabled a resistant bending moment regression with a very low mean absolute percentage error.

Algorithm	Classification	Regression
	Accuracy	MAPE
Linear Regression	-	253%
Decision Trees	0.96	2.8%
k-Nearest Neighbor	0.95	4.6%
Support Vector Machine Linear Kernel	0.96	79.2%
Support Vector Machine Radial Basis Function Kernel	**0.98**	**2.6%**
Artificial Neural Networks	**0.98**	14.7%

Table 5.32 – Classification and Regression tasks - Comparison of results

6 Conclusion

6.1 Final Considerations

The work developed throughout this thesis unveiled a small part of the large number of subjects that compose both the design process of steel connections and also the field of machine learning, enabling a deeper understanding of the key concepts behind the behaviour of steel connections and some of the most widely used learning algorithms, as well as a more general perspective of the entire process involved in the creation and validation of the models used to solve different problems in general and the problem of designing unreinforced welded beam-to-column joints in particular.

The main methods used currently for the design of beam to column connections were identified and a special attention was given to the method followed by the European standard EN 1993-1-8, the component method. A thorough description of the different components involved in this type of connection and which influence its behaviour was presented as well as the different equations used to obtained their respective resistance.

A small introduction to Artificial Intelligence in general and Machine Learning in particular, as well as the different types of learning and the different types of problems were presented.
Due to the importance that data pre-processing may have not only in the proper behaviour of the algorithms but also in the quality of the final results, some of the different techniques commonly used for the manipulation, treatment and enhancement of the datasets were also introduced.
Particular attention was given to the description of different learning algorithms and the metrics used to evaluate the quality of the models.

The proposed approach followed to solve the problem of designing unreinforced welded connections by means of machine learning techniques was presented, together with the adopted workflow and the procedure used to create and augment the dataset which served as the basis for the development of this work.

The graphic analysis of the dataset by means of 2D and 3D representations was followed by the presentation of the results obtained with the different algorithms and the necessary analysis, based on the relevant metrics, for both the classification and regression problems.

The algorithms used for the prediction of the conditioning component of the beam to column connection, with a wide range of complexity levels, led to very promising results, translated by the different considered evaluation metrics and especially by a high accuracy, in the range [0.96;0.98].

Special attention should be given to the regression task developed with the aim of predicting the value of the resistant bending moment, as this value is commonly one of the most important in the design process of steel connections. Unlike what was observed for the classification task, some learning algorithms such as Linear Regression or Support Vector Machines with Linear Kernel yielded poor results with very high mean absolute percentage errors, and thus not appropriate for practical application. However, there were other algorithms, with different levels of complexity such as Decision Trees and Support Vector Machines with Radial Basis Function Kernel that led to mean absolute percentage errors smaller than 3%, a promising result for the integration of this workflow in the different stages of real projects.

6.2 Future Developments

The analysis developed in the current work was based upon a dataset that was created with specific assumptions regarding both the adopted steel grade, the nature of the forces acting on the two elements being connected and their geometry.

Future developments of this work may consider an enlarged dataset comprising not only S355 steel graded elements but also other steel grades and a combination of beams and columns with different steel grades. Together with the consideration of different steel grades, the adoption and application of the statistical distribution of the corresponding yield stresses, in an analogous manner as was adopted for the steel sections dimensional distribution, may lead to a rapid increase in the number of examples comprising the dataset.

Another aspect is the consideration of axial forces acting on the beam and column as well as the combination of axial forces and bending moments on both the beam and column elements, leading to examples that may be closer to real cases.

Although the work here developed focused on the more standard H and I shaped steel cross sections, other databases may be created considering other cross sections such as rectangular or circular hollow sections. It may also be of interest to expand the scope of this work in order to include not only welded connections but also bolted ones.

References

Alpaydin, E. (2014). “Introduction to Machine Learning”. The MIT Press. Cambridge, Massachusetts.

Buitinck, L., et all. (2013). “API design for machine learning software: experiences from the scikit-learn project”.

Burkov, A. (2019). “The hundred-page machine learning book”.

Chapelle, O., Schölkopf, B., Zien, A. (2006) “Semi-supervised Learning”. The MIT Press. Cambridge, Massachusetts.

Comite Europeen de Normalisation. (2010). “Eurocode 3_ Design of steel structures – Part 1-1: General rules and rules for buildings”. Bruxels, Belgium.

Comite Europeen de Normalisation. (2010). “Eurocode 3_ Design of steel structures – Part 1-8: Design of joints”. Bruxels, Belgium.

Goodfellow, I., Bengio Y., Courville, A. (2016). “Deep Learning”. The MIT Press. Cambridge, Massachusetts.

Hastie, T., Tibshirani, R., Friedman, J. (2013). “The Elements of Statistical Learning: Data Mining, Inference, and Prediction” Springer New York. New York.

Hosmer, D. W., Lemeshow, S., Sturdivant, R. X. (2013) “Applied Logistic Regression”. Wiley. New Jersey.

Japkowicz, N., Shah, M. (2011). “Evaluating Learning Algorithms: A Classification Perspective”. Cambridge University Press. New York.

Jaspart, J. P., Weynand, K. (2016). “Design of Joints in Steel and Composite Structures”. Wiley. New Jersey.

Kuhn, M., Johnson, K. (2018) “Applied Predictive Modeling”. Springer New York. New York.

Mitchell, T. M. (1997). “Machine Learning”. McGraw-Hill. New York.

Montgomery, D. C., Peck, E. A., Vining, G. G. (2015). “Introduction to Linear Regression Analysis”. Wiley, New Jersey.

Murphy, K. P. (2012). “Machine learning: A Probabilistic Perspective”. The MIT Press. Cambridge, Massachusetts.

Nilsson, N. J., (2005). “Introduction to Machine Learning: An Early Draft of a Proposed Textbook”. California.

Pedregosa, F, et al. (2011) “Scikit-learn: Machine Learning in Python”. JMLR 12, pp 2825-2830

Raschka, S., Mirjalili, V. (2019) “Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow”. Packt Publishing. Birmingham, UK.

Russel, S. J., Norvig, P. (2016). “Artificial Intelligence : A Modern Approach”. Pearson, Boston.

SCI/BCSA. (2013) “Joints in Steel Construction – Moment-Resisting Joints to Eurocode 3”. The Steel Construction Institute and The British Constructional Steelwork Association. London, UK.

Silva, C., Ribeiro, B. (2018) “Aprendizagem Computacional em Engenharia”. Imprensa da Universidade de Coimbra. Coimbra.

Sutton, R. S., Barto, A. G. (2018) “Reinforcement Learning: An Introduction”. The MIT Press. Cambridge, Massachusetts.

Weisberg, S. (2013). “Applied Linear Regression”. Wiley. New Jersey.

Witten, I. H., Frank, E., Hall, M. A., Pal, C. (2016). “Data Mining: Practical Machine Learning Tools and Techniques”. Elsevier Science. Massachusetts.

Wolpert, D. H. (1997) “No Free Lunch Theorems for Optimization”, IEEE Transactions on Evolutionary Computation 1. 67

Zheng, A., Casari, A. (2018) “Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists”. O’Reilly Media. California.