UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE INFORMÁTICA

Using compression for profiling rheumatoid arthritis disease progression through data mining techniques

Diogo Henrique Rodrigues Azevedo

Mestrado em Ciência de Dados

Dissertação orientada por:
Prof. Doutor Andre Nuno Carvalho Souto

2022

Acknowledgments

To begin, I would also like to express my heartfelt gratitude to my supervisor, Prof. André Souto for all the guidance, feedback and sound advice as well as for all the weekly meetings that helped me keep motivated. To professor Alexandra Carvalho for all the feedback provided and interest shown. To the Project PREDICT (PTDC/CCI-CIF/29877/2017), funded by Fundo Europeu de Desenvolvimento Regional (FEDER), through Programa Opera-cional Regional LISBOA (LISBOA2020), and by national funds, through Fundac ao paraa Ciˆencia e Tecnologia (FCT), for providing the data for this work. A special thanks to Prof. Teresa Henriques for the keen insights on compression algorithms that helped us develop the encoder for our python module. I would also like to thank Prof. Tiago Guerreiro for expressing interest in the work developed in this thesis and for providing the opportunity to continue our studies.

To Mario Oliveira and Marta Gomes, for not letting me doubt myself and for giving me, amidst a lot of other things, the confidence further to peruse my academic studies. To everyone in ”Lá” group, Afonso Assunção, Bruno Santos, Carolina Ramos, Constança Garcia, Francisco Madruga, Guilherme Vieira, Leandro Ribeiro, Lucas Silva, João Rodrigues, João Nogueira, José Carolino, Pedro Nogueira, Primo, Rodrigo Cassanheira and Rodrigo Penedo for all the great moments and support. Last but certainly not least, to my family, for always encouraging me, and especially to my parents and little sisters, thank you for your unconditional love and support.

This work was supported by FCT through project PEI 8 BI, ref. 718, and the LASIGE Research Unit, ref. UIDB/00408/2020 and ref. UIDP/00408/2020

To my mother.

Resumo

A espondilite anquilosante é uma doença inflamatória autoimune pertencente ao ramo da espondiloartropatia das doenças reumáticas, ramos este caracterizado por doenças altamente debilitantes e com um impacto elevado tanto física como mentalmente nos pacientes que aflige. O leque de critérios para a escolha do tratamento biológico certo para cada paciente é limitado e ainda pouco claro do ponto de vista clínico o que faz com que esta escolha seja feita com base noutros aspetos, como a disponibilidade que o paciente tem para a administração a cada um dos tipos de tratamento.

Neste trabalho utilizamos aglomerados por compressão, uma abordagem baseada em teoria de informação algorítmica, na qual não existem parâmetros específicos correspondentes a área de estudo em causa, nem são necessários quaisquer conhecimentos a priori a cerca dessa mesma área. Clustering por compressão tem por base o conceito de complexidade de Kolmogorov (CK) que define a complexidade de um objeto pelo tamanho do programa mais pequeno que quando executado, produz esse mesmo objeto como output. Este conceito incomputável é utilizado para definir uma distância de informação e a sua respetiva versão normalizada, que por sua vez podem ser adaptadas para algo computável no mundo real, substituindo a CK dos objetos pelo seu tamanho após ter sido comprimido com um compressor usual. Esta versão computável da distância de informação normalizada é denominada por distancia de compressão normalizada, e é com ela que são criadas as matrizes de distancia que servem de base para a criação dos modelos de aprendizagem automática deste trabalho.

O principal objetivo deste trabalho foi, com estes modelos, estabelecer padrões entre pacientes e tratamentos biológicos que possam ajudar os profissionais de saúde a fazer uma escolha mais informada sobre que tratamento a recomendar a cada paciente.

Para a aplicação desta abordagem foi feito o estudo do estado da arte de ferramentas para clustering por compressão, no qual identificamos a ferramenta Complearn como a mais apropriada para a abordagem pretendida. O Complearn é uma ferramenta da linha de comandos do sistema operativo linux que permite a criação de matrizes de distância utilizando com a opção de escolha de diferentes métodos de compressão, bem como alguma possibilidade de customização destes mesmos métodos. Para além da computação das matrizes de distância, o Complearn disponibiliza o método dos quartetos como forma de criar grupos a partir da matriz previamente computada.

No nosso trabalho utilizamos o Complearn como ponto de partida para o desenvolvimento da ferramenta Zgli, um módulo que permite ao utilizador realizar clustering por compressão utilizando python. Neste módulo foram adicionados novos algoritmos de compressão aos já existentes no Complearn, como por exemplo, a opção de comprimir ficheiros tabulares por coluna, e um codificador para dados categóricos.

Para a criação de modelos nao usamos diretamente o método dos quartetos por ser NP-completo e portanto computacionalmente ineficiente, optando por utilizar os algoritmos *k-medoids* e *hierarchical clustering* para a criação dos modelos. Os modelos são criados utilizando matrizes de distância de compressão normalizada, que são posteriormente avaliados usando a *v-measure* e a *adjusted random score* bem como as suas matrizes de contingências e a distribuição das características de cada grupo.

No que diz respeito aos resultados foram feitos avanços em duas direções. O primeiro conjunto de resultados sendo referente a metodologia utilizada e o segundo aos padrões encontrados entre tratamentos e pacientes usando dados especificos da base de dados Reuma.pt.

No segundo grupo referente aos resultados para clustering por compressão, verificamos que o o método de compressão por coluna para os dados tabulares suplantou o método normal de compressão, criando consistentemente modelos com *scores* melhores. Verificamos também que modelos de clustering por compressão são capazes de competir com os modelos de clustering tradicionais quando aplicados ao dataset deste trabalho, obtendo *scores* semelhantes aos modelos tradicionais e fazendo-o utilizando um numero inferior de grupos.

No que diz respeito aos padrões encontrados entre pacientes e tratamentos, verificamos que INFLIXIMAB, GOLIMUMAB e ETANERCEPT tinham uma taxa de sucesso maior para pacientes do sexo masculino do que pacientes do sexo feminino, algo que é corroborado pela literatura, com o tratamento ADALIMUMAB sendo o único que apresentava uma taxa de sucesso semelhante para ambos os sexos. Verificamos também que quanto maior o ASDAS inicial, maior a probabilidade de o tratamento ser bem-sucedido, e que o pacientes com os ASDAS inicial no intervalo e 2.1 a 2.7 apresentavam uma alta chance de o tratamento falhar.

Palavras-chave: Kolmogorov, Complearn, Zgli, Clustering, AI

Abstract

Ankylosing spondylitis (AS) is a chronic autoimmune inflammatory condition belonging to the spondyloarthropathy category of rheumatic diseases characterized by being highly debilitating diseases and having a high impact on patients physical and mental health as well as social and quality of life. Biological treatment for this pathology is difficult to pick and lacks clear selection criteria. Usually, treatment is chosen based on patient convenience.

Our goal is to use an approach based on algorithmic information theory, without any domain-specific parameters to set, or any background knowledge required (clustering by compression), iterate over the current state of the art, so it can be better integrated into python pipelines as well as better suit our specific problem, and apply it to our data comprised of patients with AS so patterns between biological treatments and patient profiles can be established thereby helping clinicians make a better treatment choice for each patient.

Unsupervised clustering models are generated using normalized compression distance matrices, which are then evaluated using v-measure, adjusted random score, and visually analyzed taking into account model contingency matrix and feature distribution per cluster.

Possible patterns between biological treatment success and patient profiles were identified. Furthermore, we observed that the compression by column developed and implemented in this new tool for clustering by compression seemed to yield better results than the previous approach.

Keywords: Kolmogorov, Complearn, Zgli, Clustering, AI

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this introductory chapter, we describe the challenge at hand, and provide a clear description of the goals to be achieved, and the contributions generated throughout the making of this work. Furthermore, an outline of the thesis structure is presented.

1.1 Motivation

Rheumatic diseases [10] are chronic pathologies representing the leading cause of disability in developed countries and consuming significant financial, health, and social resources. These diseases have afflicted humankind as far back as Ancient Egypt [19], with studies showing that at least three pharaohs of the 18th and 19th dynasties of Old Egypt had ankylosing spondylitis.

Ankylosing spondylitis [29] (AS) is an autoimmune inflammatory condition belonging to the spondyloarthropathy category of rheumatic diseases. Its main early symptoms are back pain and early morning stiffness, and when left unchecked, these pains can extend to the whole spine and sacroiliac joints, with severe cases ending with the total fusion between the joints and bone structures in these areas. Most patients are young workers and AS can result in significant socioeconomic hardship because patients must take time off work and, in severe situations, may have to quit their job.

Since there is no cure for AS, treatment focuses on symptom relief and slowing down the disease progression. Treatment for this condition can go from lifestyle changes, physiotherapy, drug treatments, and the most recent, biological-treatments. Even though lifestyle changes and physiotherapy are great ways of maintaining patients' mobility and posture, therapy selection follows no specific criteria, being hard to establish what patients benefit more from each drug or biological treatment. In particular, a better understanding of therapy responses for these patients and identifying the predictors that affect therapy responses would be a highly useful piece of information.

Although the automatic based decision for Rheumatic diseases is a difficult task since, as mentioned above, even experienced clinicians may disagree on the best treatment to apply for the same patient or patient with similar characteristics, with the advent of electronic medical records, the data collected can be used to extract insightful knowledge toward the development of automatic based decision helpers. Data mining is a fast-growing field aiming to develop techniques to structure the necessary information. Cilibrasi, and Vitányi [13] proposed a new approach, based on [25], to analyze data that does not require substantial knowledge about the context of the data to produce results. This approach uses a metric defined on the information contained in the data to cluster the similarities among the instances hierarchically.

1.2 Objectives

This thesis has two primary objectives. The first was to analyze and comprehend the current state-of-the-art tools for clustering by compression, primarily Complearn, and iterate over them, adding new features that may be useful for future works and others suited to better assist us in solving the specific problem of this thesis. This new iteration will then be available as a Python module for easy integration into other Python workflows and all relevant documentation.

The second objective of this work is to apply this new tool to AS patient data retrieved from Reuma.pt, and verify the viability of this method by using it to find patterns between biological treatments and patient profiles. We will use compression methods to generate the starting distance matrices that we will, in turn, used to create plectra of unsupervised clustering models. Finally, from all the models, the best clusters are analyzed, and possible patient profiles and features of importance will be extracted and presented.

Another goal to be achieved in parallel to the first main goal is to preprocess the data, deal with noisy and missing values, and obtain a dataset that could be used for future works on the subject.

1.3 Contributions

The core contributions of this thesis are:

- Development of a state-of-the-art tool for clustering by compression [7], which can be used in future research works.

- Exploration of different compression strategies for tabular, non-tabular, and time-series data.

- Development of a dataset resulting from the merge between two other raw datasets extracted from Reuma.pt, which can be used in future research works.

- Identification of potential features of importance and patient profiles related to treatment success and failure.

1.4 Thesis outline

The rest of the document is organized as follows:

- In Chapter 2, we delve into the related work, looking at publications dealing with spondyloarthritis, the method of clustering we will use, and some of its applications in various fields.
- In Chapter 3, the background needed for the execution of the proposed work is provided. This includes an explanation of spondyloarthritis and its different assessment and evaluation methods. We present clustering by compression, the normalized compression distance, and the various steps needed to reach it. Regarding data, we look at different ways of compressing it and a method for feature selection. Lastly, we have an overview of clustering, its applications, and three different scores used in this work to evaluate its models.
- Chapter 4 describes the Complearn tool and gives an overview of its features, followed by the presentation of Zgli, a python module developed throughout this work to iterate on the Complearn tool and suit the unique demands for tackling the challenges in this study.
- Chapter 5 presents all the steps done during data preprocessing and the datasets resulting from it.
- In Chapter 6, we present the overall model-building approach and data analysis done in this work, as well as the results derived from all the datasets generated in Chapter 5.
- Chapter 7 outlines the conclusions achieved in this work and suggests topics for future work.

Chapter 2

Related Work

Clustering techniques are very useful tools for stratifying data into different homogeneous groups, and their potential to identify different patient profiles within clinical data has been shown throughout the years. For instance, in [41], the authors perform a hierarchical clustering analysis on seventy-seven Parkinson's Disease (PD) patients and fifty control individuals, identifying two patterns of cortical atrophy that separated PD patients from the control group. In another work [4], the authors develop a decision-making assistant for clinicians, where a hierarchical clustering model is used on adult spinal deformity patient data comprised of hundreds of data points, with the patterns discovered have the potential to facilitate treatment optimization by highlighting which treatment patterns yield optimal improvement with the lowest risk.

The data of this study comes as data points from patients with ankylosing spondylitis in diverse stages of biological treatment. The disease is evaluated in two main ways: patient functional ability and disease activity. Functional ability is assessed by the BASFI index [11], calculated using 10 questions regarding day-to-day activities (i.e. reaching high shelves). Disease activity can be measured using two indices, these being BASDAI and ASDAS. The BASDAI index [21] measures disease activity similarly to the BASFI index, using a questionnaire consisting of 6 questions: 2 related to pain, 3 related to affected body regions 2 regarding morning stiffness and 1 regarding fatigue. The ASDAS index is the most recent index from the tree [42], and its purpose is to improve the evaluation of the AS disease activity. This index improves on the BASFI index by being more sensitive to change and by covering disease activity more broadly.

More details of these tree indices can be found further ahead in the background chapter, but with both clustering techniques relevance and disease monitoring covered, we can now look into recent studies on data related to ankylosing spondylitis patients from Reuma.pt, that highlight the potential of the data to be mined for extraction of patterns between patients and the different biological treatment options, as well as features of importance that may describe treatment success and failure.

In another work [36] the authors employed a statistical analysis and Uni and Multivariable regression to determine baseline predictive factors of response to biological therapies at the 12 weeks. The findings show that a better response is expected in males and younger patients; baseline disease activity predicts a response at 12 weeks; baseline elevated CRP is an important predictor of ASDAS response, but not of BASDAI response; the ASDAS predicts both ASDAS and BASDAI response and the BASDAI only predicts the BASDAI response.

In another work [8], Carolina *et al.* used a joint model approach to determine the failure of biologic treatment, considering both baseline and time-varying variables from spondyloarthritis patients. Patients who were male, HLA-B27 positive, and had a later biologic therapy starting date or a longer time interval between the onset of the disease and the introduction of the first biologic therapy had higher therapy adherence.

Kishan Rama, *et al.* [35] presented a new approach to analyzing switches between therapies occurring during the patients' lifetime using AliClu, a computational method to analyze longitudinal data that provides clusters by unraveling switch patterns present in the clinical data. The clusters found allowed the authors to stratify the patients based on their temporal profiles and identify common features for those groups.

The clustering method we envisage use is based on the [13], proposed by Vitanáyi et al., and has been demonstrated to apply to several data types, including time-series data.

Music clustering [15] is one of the first examples of this method's application by the same authors. In this study, music data is clustered in different levels of granularity using the normalized compression distance and the quartet tree method, grouping the pieces of music by genre first and later grouping those same pieces of music by the author.

A recent example of this type of clustering is present in [14] where the study on the *Phylogeny of the COVID-19 Virus SARS-CoV-2*, is presented. This work uses RNA sequences of viruses as the dataset, grouping similar sequences together to find the most plausible origin of the virus.

Finally, in [39] shows an application of this method to time-series clinical data, where the same pipeline is used to analyze a dataset of fetal heart rate tracings acquired by various fetal sensors with varying capturing frequencies, sizes, and signal loss. Despite these discrepancies, the approach proved to be robust and yielded interesting results, clustering similar groups together.

Chapter 3

Background

In this chapter, we will look at the ankylosing spondylitis disease to better understand the motivation for this work and, in due course, the features represented in the data. Next, we describe the feature selection method chosen for this work, with a theoretical overview and a simple example in practice. Next, it presents the essential background on clustering, specifically clustering by compression, where we explain what normalized compression distance is, how it is computed, and all the steps needed to reach its final formula and its practical usage. Later we look into two compression approaches (integer and float compression) that were tested during the clustering development by the compression python module. Finally, we discuss the clustering technique developed in the Complearn tool, which we used to acquire testing results for our new clustering by compression Python module.

3.1 Ankylosing Spondilitis

To understand what Ankylosing Spondylitis is, let us first look at it with a broader point of view. Starting with rheumatic diseases, these are auto-immune inflammatory diseases that cause the patients' immune system to attack its joints, muscles, and bones [24]. Inside the major group of rheumatic diseases, we have the spondyloarthropathy (SpA) branch, represented by psoriatic arthritis, reactive arthritis, arthritis associated with inflammatory bowel disease, and ankylosing spondylitis (AS). These diseases share similar clinical features and an association with the human leukocyte antigen (HLA)-B27[29].

There is some nuance regarding the nomenclature and diagnosis of patients with AS. The definite diagnosis can be delayed as radiographic changes may take up to a decade to manifest. This early stage of the disease is referred to as non-radiographic axial spondyloarthritis (nr-axSpA). Disease progression will depend on many factors such as time, disease severity, and treatment response, so it is not implied that all patients with nr-axSpA may reach the more advanced AS state. The literature debates whether these illnesses should be classified as separate diseases altogether [30, 43], nevertheless axial SpA is

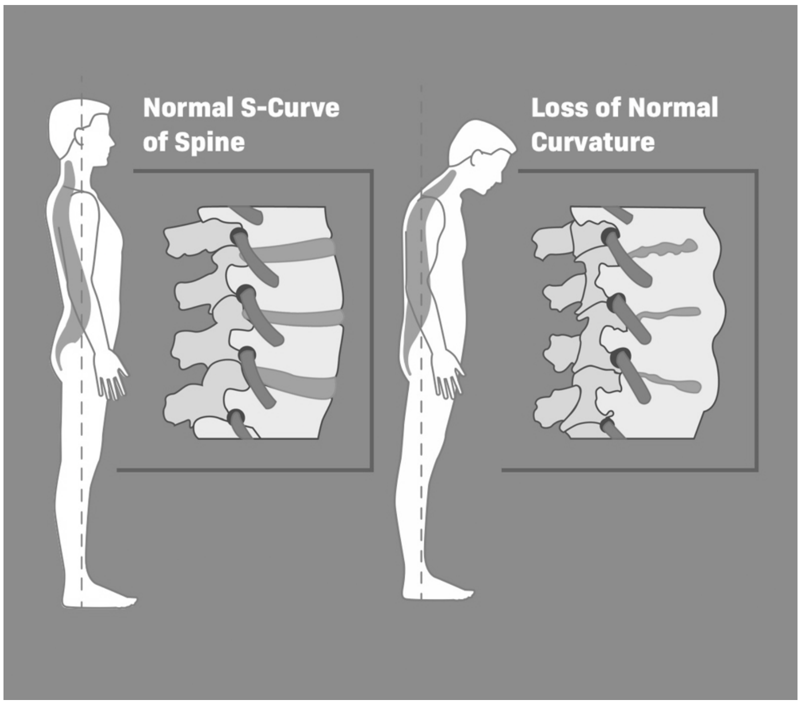


Figure 3.1: Bambo Spine. Replicated from AnkylosingSpondylitis.net [40]

nowadays the most common way to refer to these diseases [23].

Chronic inflammation, primarily affecting the axial skeleton, is a hallmark of AS. The most common symptom of AS is sacroiliitis. However, peripheral joints can also be affected. Back pain and increasing spinal rigidity are the two most common clinical symptoms. In more severe cases, the inflammation can cause fibrosis and calcification of the spine, resulting in the so-called ”bamboo spine,” which is an irreversible disorder characterized by total spinal fusion - see Figure 3.1.

These clinical manifestations can lead to systemic symptoms, including exhaustion, weight loss, chronic pain, and immobility, which can significantly influence a patient’s social and economic life. Furthermore, they can be catalysts of psychological issues like anxiety and depression, which can further reduce the patient’s overall quality of life.

Clinical disease monitoring is critical for understanding illness progression and evaluating patient response to therapy and guiding therapeutic decisions. Erythrocyte sedimentation rate (ESR), C-reactive protein (CRP) levels, which are inflammatory indicators, and other laboratory data that are thought to reveal if meaningful abnormalities are among the tests performed along side self administered questionnaires.

To evaluate the patient perception of well-being, disease is evaluated in two main ways: i) patient functional capability and ii) disease activity. The first measures to which degree the condition has impaired the patient’s movement and overall functional capability, and the latter measures the level of symptom frequency and intensity.

Starting with function capability, we have Bath Ankylosing Spondylitis Functional

Index (BASFI)[2], calculated using patient answers from 0 to 10 to ten questions related to the difficulty of daily activities, such as bending at the waist, reaching for high shelves, going upstairs, standing up without external help, etc... To obtain the final index, every function is given the same weight by performing the mean of every answer and the result is a number between 0 to 10.

For disease activity measurement, we have Bath Ankylosing Spondylitis Disease Activity Index (BASDAI)[5] and the more recent Ankylosing Spondylitis Disease Activity Score (ASDAS)[28]. Starting with BASDAI, it is calculated using patient answers from 0 to 10 to six other questions related to disease symptoms. These questions cover fatigue, spinal pain, joint pain/swelling, localized tenderness (enthesitis, or inflammation of tendons and ligaments), morning stiffness duration, and morning stiffness severity. To balance each symptom equally, the mean of the two-morning stiffness values are taken. The resulting score from 0 to 50 is divided by 5 to yield a final BASDAI score between 0 and 10. Scores of 4 or higher indicate sub-optimal illness control, and patients with these scores are usually suitable candidates for either a change in their medical medication or enrolment in clinical trials assessing new drug therapies.

ASDAS is a composite index that combines five disease activity variables with only partial overlap, resulting in a single score with better truth (validity), enhanced discriminative capacity, and improved sensitivity to change compared to single-item variables. ASDAS value ranges from 0 to 10, and there are two ways to calculate it, with the preferred being the following:

$$\begin{gathered}0.12 \times BackPain+\\ 0.06 \times DurationOfMorningStiffness+\\ 0.11 \times PatientGlobal+\\ 0.07 \times PeripheralPainSwelling+\\ 0.58 \times \ln(CRP+1)\end{gathered}$$

Where $\ln$ is the discrete logarithm in the natural base and CRP is the level of c-reactive protein in the patients' blood (measured beforehand by a CRP test). Further, the information about disease activity can be extracted if the index is compared with the following cut-offs:

- $ASDAS < 1.3$: Inactive Disease;
- $1.3 < ASDAS < 2.1$: Low Disease Activity;
- $2.1 < ASDAS < 3.5$: High Disease Activity;
- $ASDAS > 3.5$: Very High Disease Activity.

If, in two consecutive appointments, there is an ASDAS score decrease of 2.0 or more points, it means there was a significant clinical improvement, and if this decrease was greater than 1.1 or more points, it is consideredan important clinical improvement.

The ASAS-EULAR group uses these indices to make recommendations for the management and treatment of AS - see figure 3.2. SpA treatment should be personalized to the individual patient, considering their characteristics and symptoms, with the most usual objective being to reach a state of disease inactivity. Physical therapy, nonsteroidal anti-inflammatory drugs (NSAIDs), disease-modifying anti-rheumatic drugs (DMARDs), and biologic agents, such as tumor necrosis factor (TNF) inhibitors and interleukin (IL)-17 or IL-23 inhibitors, may be used as treatment options if patients remain in a high disease activity state after trying the referred options.

As we can see from the recommendations in 3.2, treatment choice represents the main problem for SpA treatment, since the same treatment can have radically different responses from different patients. For this reason, the challenge becomes the optimization of this trial and error process, so the best treatment is provided as early as possible, and we reduce the disease's impact on patient functional ability and overall quality of life.

3.2 Kolmogorov Complexity

The Kolmogorov Complexity (K) is a concept from algorithmic information theory, a sub-field of mathematics and computer science, and it can be described as follows a measure of information in a string. The Kolmogorov complexity of an object - in the case of this paper, text files containing patient features related to AS - is the length of the shortest program in a predetermined syntax (i.e, Turing Machine, Python, Java, Lisp) that, when run, produces that same object as output. So $K(x)$ is the length of the most compressed version describing x.

As an example that illustrates this concept, consider the following strings:

- s_1: "ggggggggggggggggg" size $= 17$
- s_2: "LATQvgkCQaNwEadqO" size $= 17$

Notice that although they have the same size in terms of algorithmic information needed to describe the two strings, one can use the following programs p_1 and p_2 describing s_1 and s_2, respectively:

- p1: "return 'g' * 17 " $K(s_1) \approx 4$
- p2: "return 'LATQvgkCQaNwEadqO'" $K(s_2) \approx 17$

Notice that the number of times 'g basically describes the information in s_1' is repeated; therefore, $K(s_1)$ (the algorithmic information needed to describe s_1) is much shorter than writing the entire string. On the other hand, there are no "patterns" that one can use to describe more concisely the string s_2; therefore $K(s_2)$ is nearly maximally, i.e., approximately equal to the size of s_2.

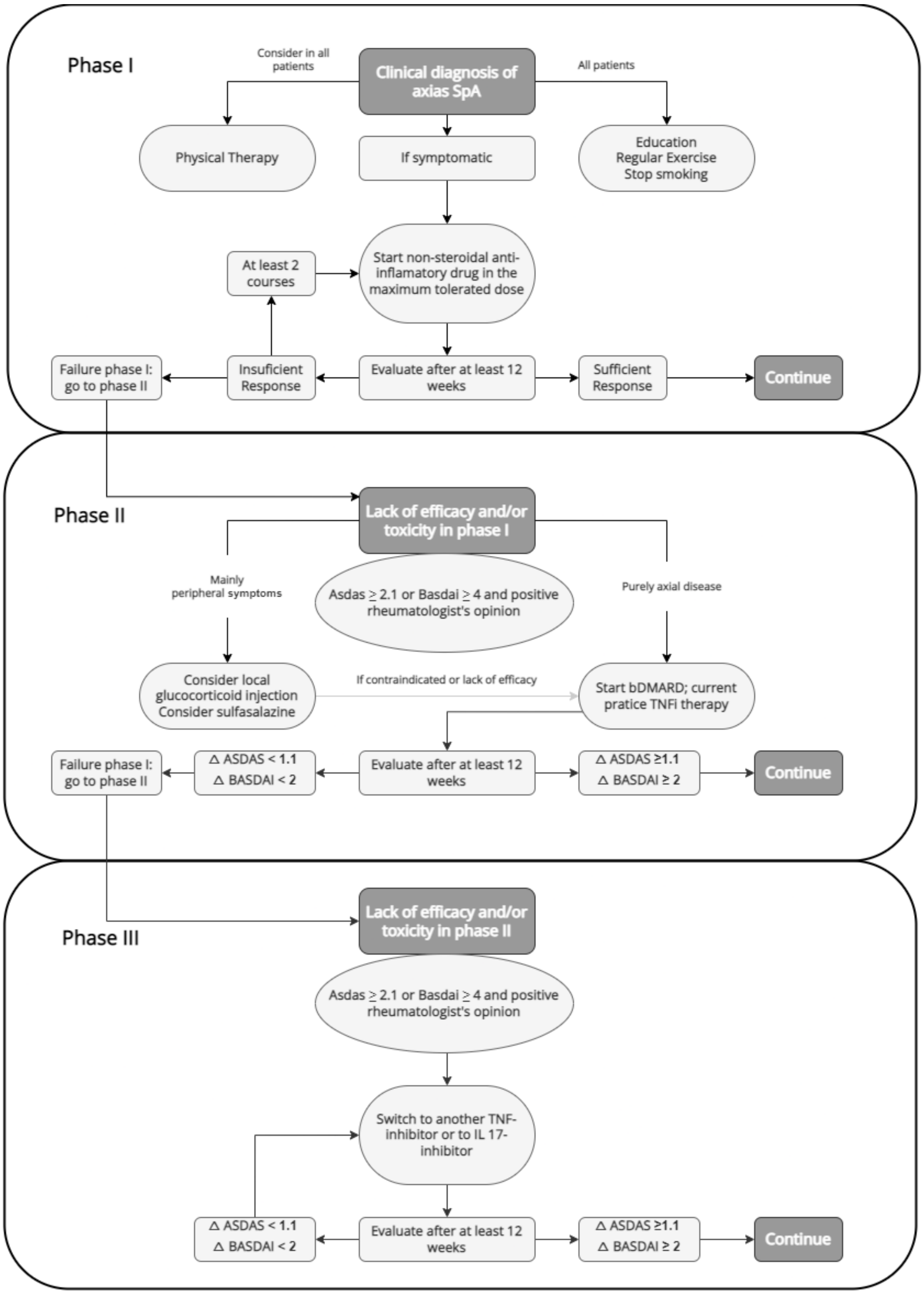

Phase I
Consider in all patients
Clinical diagnosis of axias SpA
All patients
Physical Therapy
If symptomatic
Education
Regular Exercise
Stop smoking
At least 2 courses
Start non-steroidal anti-inflamatory drug in the maximum tolerated dose
Failure phase I: go to phase II
Insuficient Response
Evaluate after at least 12 weeks
Sufficient Response
Continue
Phase II
Lack of efficacy and/or toxicity in phase I
Mainly peripheral symptoms
Asdas ≥ 2.1 or Basdai ≥ 4 and positive rheumatologist's opinion
Purely axial disease
Consider local glucocorticoid injection
Consider sulfasalazine
If contraindicated or lack of efficacy
Start bDMARD; current pratice TNFi therapy
Failure phase I: go to phase II
△ ASDAS < 1.1
△ BASDAI < 2
Evaluate after at least 12 weeks
△ ASDAS ≥1.1
△ BASDAI ≥ 2
Continue
Phase III
Lack of efficacy and/or toxicity in phase II
Asdas ≥ 2.1 or Basdai ≥ 4 and positive rheumatologist's opinion
Switch to another TNF-inhibitor or to IL 17-inhibitor
△ ASDAS < 1.1
△ BASDAI < 2
Evaluate after at least 12 weeks
△ ASDAS ≥1.1
△ BASDAI ≥ 2
Continue

Figure 3.2: Algorithm based on the ASAS-EULAR recommendations for the management of axial spondyloarthritis.

3.3 Normalized Compression Distance

The *Normalized Compression Distance* measures the similarity between two objects, but to understand it, we first need to look at the concept of information distance and its normalized version. The information distance (ID) represents the distance between two objects x and y by the shortest program (the one corresponding to Kolmogorov Complexity), which transforms x into y and vice-versa.

$$ID = \max\{K(x|y), K(y|x)\}.$$

The information distance will then give us the absolute distance between two objects independently of the size of the objects, but if we want to measure similarity between objects, we must first normalize this distance to get the relative distances between objects. For example, the ID can tell us that two strings differ by 17 bits, but does not take into account if this difference is between two strings of size 50 or 500. For this reason, we divide the information distance by the size of the most extensive description of the two strings.

$$NID = \frac{max\{K(x|y), K(y|x)\}}{max\{K(x), K(y)\}}.$$

With the NID, we can then measure the similarity between any computer file. However, since this distance is calculated based on the representation of each object with the least Kolmogorov Complexity, and this notion, in turn, is not computable, we have to use an approximation of this measure: the Normalized Compression Distance (NCD).

The Normalized Compression Distance approximates of K by using real-world compressors Z (i.e. *gzip*, *bzlib*, *zlib*), to represent the size of compressed objects. Having this notion in mind, the NID was rewritten to be applicable in the real world [26]:

$$NCD = \frac{Z(xy) - min\{Z(x), Z(y)\}}{max\{Z(x), Z(y)\}}.$$

The distance is calculated by first appending both files together and compressing them. Then both files are compressed individually, taking the difference between the appended compressed file size and the smallest individual compressed file size. Finally, this difference is normalized by dividing it by the size of the largest individual compressed file. Since we are using compressors Z to approximate K of each object, it is obvious that the better the Z is, the more accurate the NCD results will be.

3.4 Integer Compression

As we have seen before, data compressors will be essential in this work to accurately approximate each object's Kolmogorov version. Since the data are written in files representing time-series of integer variables, here we look at some methods that compress these integer files without forgetting to consider their time-series structure.

Delta and Delta-of Delta encodings [27] compresses files by only storing the difference (delta) of the last iteration of that same object. This method works best when there is repeated information in succession. So, for example, given the data:

Appointment	BASDAI	ASDAS	BASFI
2021-10	8	34	8
2021-11	7	28	7
2021-12	4	18	4

We can apply Delta and we obtain:

Appointment	BASDAI	ASDAS	BASFI
2021-10	8	34	8
1 month	-1	-6	-1
1 month	-3	-10	-3

We can then go one step further and apply Delta encoding to the already compressed data to compress specific columns even further. This is called Delta of Delta encoding. In the following example, we know that the appointment column has a frequency of 1 month, which means that the delta value will be repeated in every row of the table. The delta of this specific column can then be performed to compress the data even further. The first and second rows stay the same storing the starting rows of the first and second delta encoding, respectively, all the others turn to zeros since there are no other differences in the column.

Appointment	BASDAI	ASDAS	BASFI
2021-10	8	34	8
1 month	-1	-6	-1
0	-3	-10	-3

After encoding the data with this method and reducing the number of digits needed to represent it, we can still go one step further and use the least memory-heavy way of storing the integer data inside the files.

3.5 Float Compression

In general, compressing floating-point values is more challenging than compressing integers. Unlike fixed-length integers, floating-point numbers frequently employ all their available bits, especially when converted from decimal numbers that are not properly represented in binary. This is because floats are built from base 2 fractions that do not entirely overlap, so decimal values are rounded to the nearest base 2 value available. An example could be when we add 0.1 to 0.2 we do not get 0.3 and get 0.30000000000000004 instead.

Also, delta-encoding techniques do not work well on floats, as they do not reduce the number of bits necessary to represent its value. Because of these difficulties, most floating-point compression algorithms tend to be complex, slow, or lossy (e.g, truncation of significant digits).

One of the fastest and lossless float compression techniques is the XOR-based Algorithm used in the Gorilla [34] in-memory time-series database developed by Facebook.

This technique consists of the XORing of successive floating-point values. It starts by looking at the hexadecimal values of the floats, (1) storing the first hexadecimal value without changing its value, then (2) we perform the XOR of the second hexadecimal value with the first and storing its value, (3) finally, it repeats this process for the remaining values. The following tables illustrate this process:

Float	Hex Representation	XOR with previous
15.5	0x402f000000000000	
14.0625	0x402c200000000000	0x0003200000000000
3.25	0x402a000000000000	0x0026200000000000
8.625	0x4021400000000000	0x002b400000000000
13.1	0x402a333333333333	0x000b733333333333

The Quartet Method

Now that we have established the various methods for approximating the NCD using compressors, whether they are general-purpose compressors like GZIP or ZLIB, or more specialized compressors like int and float compression, we now describe a way to group the objects and visualize these clusters once the NCD matrix has been calculated.

We describe the quartet method as, from our collection of n elements, we consider each group of four items. Therefore, the total number of groups is $\frac{n}{4}$. A tree is then built from each group u, v, w, x, and each internal node has three neighbors, implying that the tree is made up of two subtrees with two leaves each. Let this tree be called a quartet. There are three possible dispositions for a quartet: $uv|wx$, $uw|vx$, and $ux|vw$, where each pair of letters represents the leaves connected to the same internal node as shown in Figure 3.3. The cost of each quartet is then defined by the sum of distances between each pair of neighbors: $C_{uv|wx} = d(u, v) + d(w, x)$. We say a tree T is consistent with u, v, w, x if and only if the path from u to v does not cross the path from w to x for any given T and any collection of four leaf labels u, v, w, x. The tree representing our data can then be considered composed of this smaller quartet tree. Finally, the sum of the costs of all consistent quartets is the cost of the entire tree.

The authors of [13] go even further and implement a score S to measure how well any given tree represents the pairwise distance matrix. For this, a list for all quartets of all groups is created. Then, for each group, the best and worst scores are calculated (i.e, maximal and minimal distances). Adding all the best scores will yield the best theoretical score a tree could have for a given matrix, and its opposite could be done to calculate the worst score. To compare tree scores, we rescale the best and worst scores to be 1 and 0, respectively.

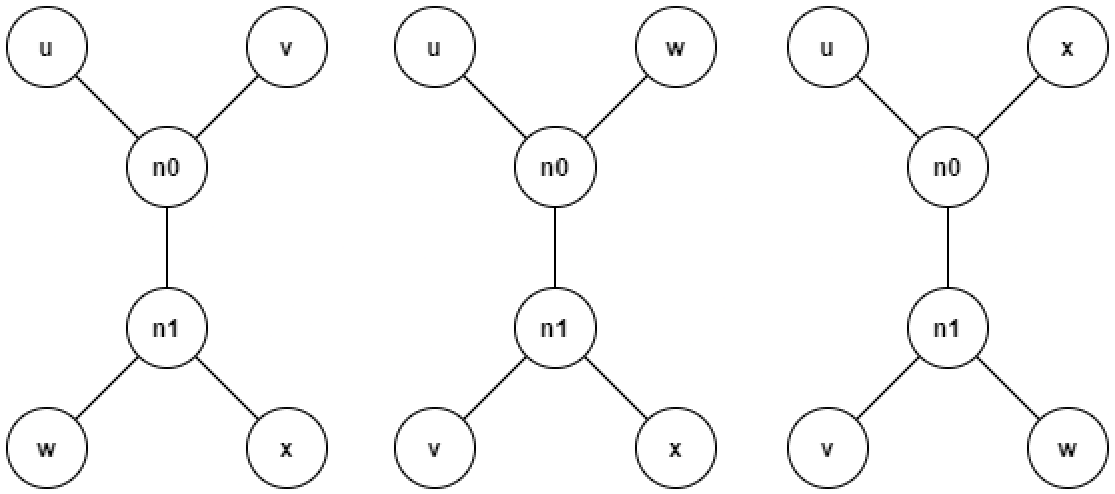


Figure 3.3: Possible position of quartets of 4 nodes.

A method based on randomization and hill-climbing is used to find an optimal tree. The search method starts by creating a random tree T_1 with $n-2$ internal nodes and n leaf nodes ($2n-2$ nodes total). The score $S(T_1)$ is then calculated, and since it is the first score calculated, it denotes the best-known score up until that moment. A second tree T_2 is then created by mutating T_1 in one of three possible ways: (1) a leaf swap, choosing two leaf nodes randomly and swapping them. (2) a subtree swap, choosing at random two internal nodes and swapping the subtrees rooted on them; (3) a subtree transfer, choosing a random tree at random, detaching and reattaching it in another place, maintaining similarity invariants. After T_2 is created, we calculate its score $S(T_2)$. If $S(T_2)$ ¿ $S(T_1)$ then we keep the tree T_2, otherwise, we keep T_1 and repeat the process. Finally, the search algorithm halts if it reaches a tree with a score $S(T) = 1$ or if no better trees are being found in a reasonable amount of time.

3.6 Maximum Relevance Minimum Redundancy

When building a machine learning model, it is clear that the more data we have to train a model, the better its performance is, but this applies only to rich information data. For instance, in [16], the authors show that, from a dataset with thousands of features, the best accuracy results for predicting a disease are obtained using a much smaller subset of features (say, 50). Therefore, feature selection procedures are critical for achieving better model performance.

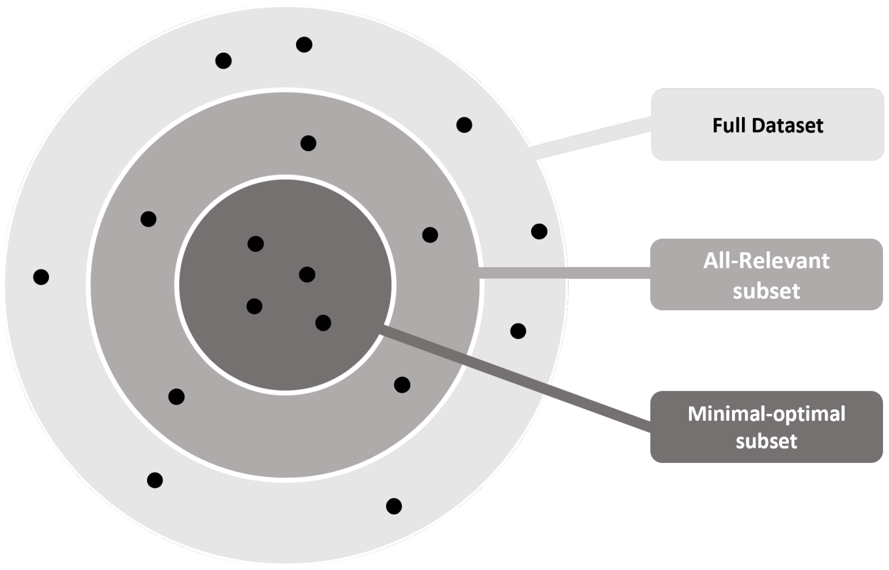


Figure 3.4: Feature selection subsets

Feature selection can be done in two main ways:

- All-Relevant methods aim to find all of the features that have any predictive potential.
- Minimal-Optimal (such as MRMR) methods that aim to identify a small number of features that, when combined, offer the most significant predictive power.

In practice, this means that if 2 features have the same predictive score, both would be selected by an all-relevant method, whereas only one would be selected and the other dismissed by a minimal optimal method. This is a subtle difference when looking at small datasets, but as we scale up the main dataset, the number of features between the all-relevant dataset and the minimal optimal dataset becomes very clear, making the algorithm more efficient while keeping model performance.

In the work developed, we used MRMR as the tool for feature selection. This approach works as follows: The first step is to choose the number of features K we intend to extract from the main dataset.

Next, a basket of K size is created, and a feature is selected and added to it at the end of each iteration until the basket is full (number of features in basket = K). One can easily understand the mechanism for the selection of the features if we interpret the method's name - Maximum relevance Minimum Redundancy - in each iteration, when selecting a feature, we choose the one that is most relevant to the target variable and has the least amount of overlap with the features chosen in the earlier iterations.

This relation between relevance and redundancy is given by a score obtained using the following formula:

$$score_i(f) = \frac{relevance(f|target)}{redundancy(f|features\ selected\ until\ i-1)}.$$

The best feature of the i-th iteration would be the one with the highest score.

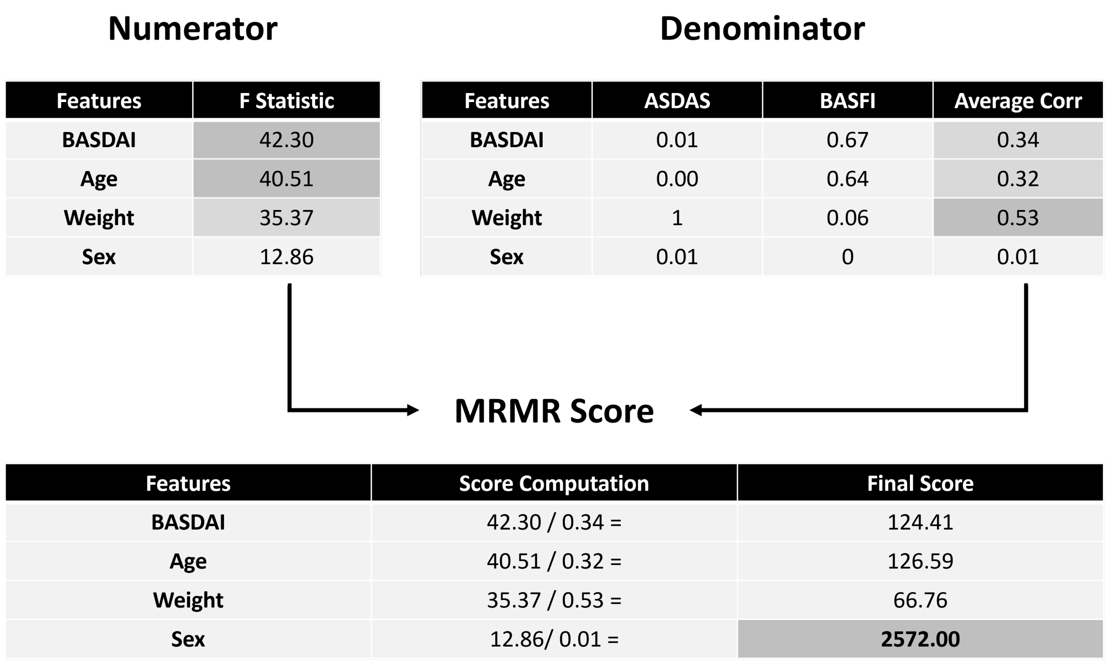
Now that we have the concept of score down, the final step is to translate the abstract concepts of relevance and redundancy into something that can be calculated. Starting with the score's numerator, we can represent the relevance of a feature by the F-statistic between the feature and the target. Looking at the denominator, we can represent the redundancy of an element by the average Pearson correlation between the feature and all the other features already in the basket. If we do this, our computable score formula becomes:

$$score_i(f) = \frac{F(f|target)}{\sum_{s \in features\ select\ until\ i-1} |corr(f,s)|/(i-1)},$$

where i is the i-th iteration, f is the feature being evaluated, F is F-statistic and corr is Pearson correlation. Note that correlation is taken in absolute value. If two features have a correlation of .9 or of -.9 it makes no difference: in both cases, they are highly redundant.

Finally, to illustrate how the method works, consider a modified example from [31] that displays a single iteration of the Algorithm. We assume the following state of the Algorithm:

- The Algorithm is currently on its 3rd iteration;
- ASDAS has been selected in the first iteration;
- BASFI has been selected in the second iteration.


Numerator
Denominator
MRMR Score

Features	F Statistic
BASDAI	42.30
Age	40.51
Weight	35.37
Sex	12.86

Features	ASDAS	BASFI	Average Corr
BASDAI	0.01	0.67	0.34
Age	0.00	0.64	0.32
Weight	1	0.06	0.53
Sex	0.01	0	0.01

Features	Score Computation	Final Score
BASDAI	42.30 / 0.34 =	124.41
Age	40.51 / 0.32 =	126.59
Weight	35.37 / 0.53 =	66.76
Sex	12.86/ 0.01 =	**2572.00**

Figure 3.5: Example of MRMR 3rd Iteration

Looking at the MRMR scores obtained at the end of the 3rd iteration, we can see that the next best feature is Sex, with a score of 2572.

3.7 Clustering

Clustering is the process of grouping a population or set of data points into groups with inter-group heterogeneity and intra-group homogeneity. This means that points within the same group are similar but distinct from points in other groups.

Clustering has applications in many fields, including market research, pattern recognition, data analysis, and image processing. As an example of this technique being applied, we can look at pattern and cluster mining on text data [3] where the authors use hierarchical agglomerative clustering and k-means to group similar text files, such as stories, emails, and news. Another example of clustering being utilized, this time in health studies, is the clustering of patients based on their individual course of low back pain over a six-month period [6] where four clusters that described patients of different rates of improvement were found ranging from slow improvers to fast improvers.

In this work we will be using K-medoids, and Hierarchical agglomerative clustering as techniques that take a distance matrix as input and generate clusters in an unsupervised manner and present them as output. We refer to Hands-On Unsupervised Learning Using Python [33] for detailed explanation as well as a hands-on approach of this methods in python.

The first is a very simple algorithm that divides n observations into k clusters, with each observation belonging to a cluster and the nearest mean acting as the cluster'centroid therefore, in the construction of the cluster the next element is placed in the group for which the distance is minimal. The second is a bottom-up approach to hierarchical clustering methods, in which all points begin in a different cluster and are joined recursively until the desired number of clusters is reached.

Clustering results can be evaluated in many different ways, but in this work we will be using the silhouette score, v-measure and adjusted random scores to look at the performance of our models.

The silhouette score [38] is usually used when the data's ground truth labels are unknown. The score values can range from -1 to 1, with -1 for incorrect clustering and 1 for highly dense clustering. The score of a single sample is given by:

$$s = \frac{b - a}{max\{a, b\}},$$

with *a* representing the mean distance between a sample and all other points in the same class, and *b* representing the mean distance between a sample and all other points in the next closest cluster. This means that when the silhouettes are dense and well-spaced, the silhouette score will be higher, corresponding to the usual definition of a cluster.

The v-measure[37] score can only be used when the ground truth labels are known. This score is based on two concepts: homogeneity, meaning each cluster contains only members of a single class, and completeness, where all members of a given class are assigned to the same cluster. Both the homogeneity and completeness values are bound between 0 and 1, with their harmonic mean being the so-called v-measure. The v-measure is computed as follows:

$$v = \frac{(1+\beta) * homogeneity * completeness}{(\beta * homogeneity * completeness)},$$

with β default value being 1. If we change β to values below 1, more weight will be given to the homogeneity of the clusters. On the other hand, if the value of β is greater than 1, more weight will be attributed to the completeness of the clusters.

Finally, we have the random adjuster score [1], which can only be used if the ground truth labels of the data are known. This score is a corrected version of the rand index that ensures values close to 0 for a model outputting random labeling of the objects. This score is symmetric, i.e., swapping the argument does not change the scores. Its values range from 0 to 1, with 0 representing random labeling and 1 representing perfect labeling.

Chapter 4

Zgli

In this chapter we will take a closer look at the Complearn tool, explain how it works and its main features. Afterwards we will explain the motivation behind the creation of the Zgli python module, as well as its main features and improvements developed for this particular work, looking at the main validation tests at the end.

4.1 Complearn

Complearn [12] is a Linux command line tool that enables its users to perform clustering by compression in a straight forward and intuitive way, by providing all the resources necessary for distance matrix computation and cluster production and visualization.

The tool is divided in two main command names, these being *ncd*, for computations of the Normalized Compression Distance and *maketree*, for binary tree generation from a given distance matrix.

The ncd command name offers the possibility to select between different compression methods among *bzlib*, *zlib* and *blocksort*, and multiple input options described below:

- File Mode - Takes as argument a filename whose contents will be compressed.
- String Literal Mode - Takes as an argument a string whose contents will be compressed. By default, each string literal is separated by white space. For string literals containing white space, surround with double quotes.
- Plain List Mode - Takes as argument a filename, which contains a list of filenames to be individually compressed. Each filename is separated by a line break.
- Term List Mode - Takes as argument a filename whose contents are a list of string literals to be individually compressed. Each string literal is separated by a line break.
- Directory Mode - Takes as argument the name of a directory whose file contents will be used to compute the distance matrix.

The outputs from the *ncd* commands, if we so choose, can then be used as input for the *maketree* command name. This command takes a distance matrix as input and computes a possible best-fitting unrooted binary tree and outputs a "treefile.dot" that can be visualized using the GraphViz tool [17] with the only two restrictions being that the matrix must be square and be of size 4x4 or greater. Internally the *maketree* method generates and represents its structure using the quartet method explained in section Float Compression.

4.2 Zgli

The Zgli [7] module was constructed to be a python iteration over the Complearn tool by porting its distance matrix computation capabilities and using it as a starting point for development and to facilitate the use of other compressors and to have the access to specific type compressors such as the gorilla compressor for float type data and delta compressor for int type data.

The tool is written in python to better facilitate its integration with other established data science utilities and being it a scripting language, development and editing of the tool becomes much easier than if it was being made using C with the Complearn tool source code as a starting point.

The Zgli is divided in two main classes:

- Folder - Perform operations inside the folder containing the files intended for compression and clustering.
- Encoder - Encode tabular data using functions to transform the data.

In the following subsections we take a look at the Folder and Encoder classes, explaining their purpose and main features but skip over description, parameter, and examples of some of the functions. For a detailed description of all the functions, code and usage as well as a quick start guide we refer to the Zgli website [7].

4.2.1 Folder

Four separate functions compose the Folder class, three of which provide the user with access to relevant information about the files and one of which enables the user to generate a distance matrix that encompasses all the NCDs between the files in the folder.

The first three functions are *get_file_names*, *get_file_lengths*, and *get_file_sizes*, where length denotes the text file's character count and size denotes the amount of memory it uses to be locally stored.

The final function in this class, *distance matrix*, computes several methods for handling the data by importing the Complearn methods to this functionality. Starting with compression options, Zgli now offers the additional *gzip* , *lzma*, and *raw* compression options in addition to the pre-existing *zlib* and *bzlib* compressors present in complearn with *raw* meaning the files are not compressed and the original file sizes are used instead for the computation of the distance matrix.

The option to compress a text file per column was the second feature we added to this function. This function will calculate the file size as the sum of the compressed sizes of all the columns inside the text file if it has a tabular structure and the user provides the column delimiter (e.g., commas in .csv files).

Additionally, the user can perform weighted compression by column, where the compressed size of each column is multiplied by the weight provided, allowing the user to artificially manipulate column importance on the final compressed file size.

4.2.2 Encoder

The Encoder class has three different functions that allow the user to encode small tabular data into larger sequences, with the idea that the text file compressors are then capable of analyzing and compressing. This class was idealized due to the necessity to solve the problem that appears when compressing small text files using state-of-the-art compression algorithms. Small files, even if completely different, when compressed will yield very similar compressed sizes, making the NCD unable to correctly depict the differences between them.

With the three encoding functions *categorize_cols*, *standardize_categorical_cols*, *encode_df* provided in this class, the user can create a pipeline and encode the smaller data that is usually present inside the cells of tabular data, and still be able to use the clustering by compression approach to the data.

For an encoding pipeline example, let's look at the following data:

Feature1	Feature2	Feature3	Feature4
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2

After loading our data, we encode all the columns comprised of continuous variables using the *categorize_cols* function. The data is then transformed to what is shown below:

Feature1	Feature2	Feature3	Feature4
(4.296, 5.2]	(3.2, 3.8]	(0.994, 2.475]	(0.0976, 0.7]
(4.296, 5.2]	(2.6, 3.2]	(0.994, 2.475]	(0.0976, 0.7]
(4.296, 5.2]	(2.6, 3.2]	(0.994, 2.475]	(0.0976, 0.7]
(4.296, 5.2]	(2.6, 3.2]	(0.994, 2.475]	(0.0976, 0.7]
(4.296, 5.2]	(3.2, 3.8]	(0.994, 2.475]	(0.0976, 0.7]

With all the columns as categorical, we now standardize the dataframe structure, so all categories are represented with the same notation, using *standardize_categorical_cols*:

Feature1	Feature2	Feature3	Feature4
0	2	0	0
0	1	0	0
0	1	0	0
0	1	0	0
0	2	0	0

Finally, using the *encode_df* function, we can encode all the categories into their representative patterns, with the final format of the data being the following:

Feature1	Feature2	Feature3	Feature4
000000000000	012012012012	000000000000	000000000000
000000000000	010101010101	000000000000	000000000000
000000000000	010101010101	000000000000	000000000000
000000000000	010101010101	000000000000	000000000000
000000000000	012012012012	000000000000	000000000000

To better understand how the *encode_df* function works, let's look at the following example, starting by establishing the following:

- Row: 0,1,0,2
- ASCII string: 0123456789abcdefg (...)
- Hop: 1

With the Row being a single row from a dataframe after the *categorize_df* and *standardize _categorical_cols* functions were applied, the ASCII string being a string made up of 94 different ASCII characters, and Hop being a user defined parameter that determines how many characters should be jumped between different categories (the purpose of this parameter should be clear by the end of the example).

Now that we've established the data and parameters, let's move on to the row to be encoded. The row is divided into three classes: 0, 1, and 2. This means that the encoding function will need to generate three distinct patterns to represent each of one of these classes. All of these patterns must have varying levels of complexity, so that when compressed, different compressed sizes are obtained. This variation in complexity must also be consistent across classes so that feature importance remains generally constant. The solution proposed by *encode_df* for all of these issues is to encode the classes as follows:

- 0: 000000

- 1: 010101
- 2: 012012

Where every class is represented by a slice of the main ASCII string, with the size of this slice increasing by one (hop = 1) for each new class. If the value of hop parameter was equal to two, the size of this slice would increase by two for each new class, with the final encoding results being:

- 0: 000000000000000
- 1: 012012012012012
- 2: 012340123401234

Another thing to note, and this is visible in the previous example, is that the size of the encoded string that represents each class grows in proportion to the number of classes and hop size. As previously stated, the trend is for larger strings to generate larger compressed sizes, hence all classes must have same size strings representing them, so the differences between them can reside in their complexity rather than their size. If we want all the strings to be the same size, we must consider the hop parameter and the number of classes the function must represent, with the size formula being:

$$size = \mathrm{lcm}(hop, hop \times 2, \cdots, hop \times n)$$

where n is the number of classes of the column with most classes in the dataframe and lcm represent the least common multiple among all the numbers given as parameters. Furthermore, to make sure all the columns are represented in a same manner, this function makes sure to distribute the class representations between all the columns in the best way it can. For example, let's take the representations of column X (column with most classes in the dataset):

- 0: 000000000000000
- 1: 012012012012012
- 2: 012340123401234

To make sure a binary column, for example column Y, has the same impact on the compression sizes as column X, the function encoded column Y in the following way:

- 0: 000000000000000
- 1: 012340123401234

This function considers the difference between class 0 and class 2 in column X is the same as the difference between class 0 and class 1 in column Y, and for this reason, tries to represent them in the same way. When this is not possible, the function rounds the class representation to the closest existing representation in the column with most features inside the dataframe, and when tied, this approximation is rounded downwards.

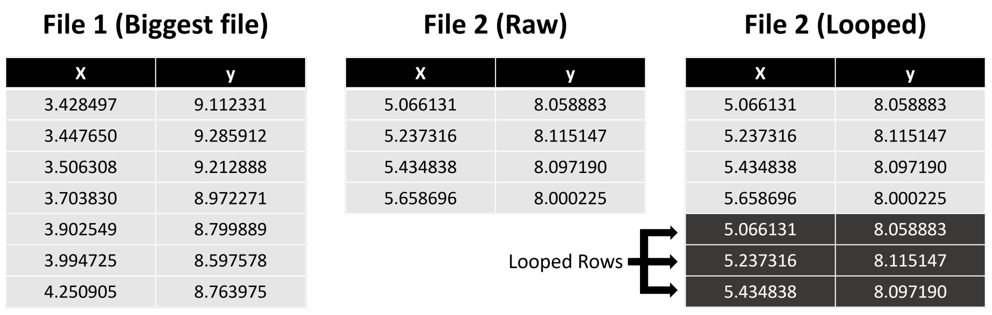
4.3 Validation Tests

In this section, we look at various questions where we compare the zgli module to the Complearn tool to see if it enhances the state-of-the-art tolls for clustering by compression in any manner other than simply being a Python port of an already existing tool. These questions are:

- Can we apply the Complearn tool to multivariate, possibly time series data? Are the results consistent?
- Can these results be improved?
- Can dedicated compression methods like int/float compression prove to aid or be better than text compressors like *gzip*, *bzlib* or *lzma*.

Due to legal constraints, access to the data took longer than expected. Therefore, some additional data was used to validate the compress by column feature of our Zgli python module. Since all we knew at the time was that our data would be mostly numeric, with float values and would probably, at least, have a subset of time-series data, we searched the UCI repository for datasets with similar features and came across this basketball dataset [22] by David Guarin et al. An accelerometer (x,y,z) and a gyroscope (R, phi, delta) located on the right arm were used to collect the data. Four distinct users were requested to carry out several basketball moves: dribble, hold, pass, pickup, shoot. All the data collected was stored as float values. Each pair of user-action as its own file, and the dataset has a total of 80 files. Little data munging was required since there were no missing values and the data was already pre-arranged in text files, the required format for our method. Given the various file sizes and the method we intend to employ, a separate dataset was built from the raw one (after the headers were removed) in which each file has the same number of lines. This was accomplished by looping the smaller files and repeating them line by line until all the smaller files were the same size as the largest one - see Figure 4.1.

To validate this method's results, it's important to take into account the following: The files will be clustered together based on their compressed size. Although not always, files with larger original sizes tend to have larger compressed sizes, and files with smaller original sizes tend to have smaller compressed sizes. As a result, we believe it is critical



File 1 (Biggest file)

x	y
3.428497	9.112331
3.447650	9.285912
3.506308	9.212888
3.703830	8.972271
3.902549	8.799889
3.994725	8.597578
4.250905	8.763975

File 2 (Raw)

x	y
5.066131	8.058883
5.237316	8.115147
5.434838	8.097190
5.658696	8.000225

File 2 (Looped)

x	y
5.066131	8.058883
5.237316	8.115147
5.434838	8.097190
5.658696	8.000225
5.066131	8.058883
5.237316	8.115147
5.434838	8.097190

Looped Rows

Figure 4.1: Looped rows example.

to examine clusters formed using only the original file sizes so that we can later compare them to clusters generated using file compressors and see what their impact is on the groups formed. After looping all the necessary files in the 80 file dataset, we sampled 15 files (3 files per basketball move) just so the binary-trees resulting from the clustering performed could be better visualized for the effect of this explanation. Figure 4.2 shows the binary tree generated when using the uncompressed file sizes, and it clearly shows that the method is not able to correctly group the actions based on the raw size alone.

Firstly we generated the tree using the original file sizes, then both compressors available in the Complearn tool, *zlib* and *bzlib* were used for clustering, with the *zlib* showing the best results. The tree differs greatly from the one using the original file sizes with almost every action being grouped correctly and the tree itself having a score of approximately 0.98 as seen in

For the second test, we wanted to look at the performance of the method when all the files had the same number of rows. So firstly, we had all smaller files have their structure repeated row by row (as illustrated before) until they reached the same number of rows contained in the largest file. The dataset resulting from this operation has noisier data since files now contain repeated rows and different files have a different number of repeated rows. Complearn was then applied to the altered dataset, and the best results did not differ much from the first test. The compressor with better performance was now *bzlib*, but the purity of the clusters found on the binary tree and the tree score did not have a significant change.

Following this analysis of the Complearn tool, we turned to our Zgli module, and started testing with the new features implemented to then compare them to the baseline established by the previous tests. After the introduction of file compression by column, the compression methods *zlib*, *bzlib*, *gzip*, *lzma*, and *gorilla* were all tested. The first four were also tested with and without *gorilla* encoding beforehand, and some interesting outcomes were discovered. Firstly, the *gorilla* encoding before compression did not help in the grouping of the files, in fact, all the binary trees where the float encoding was applied showed both worse scores and clusters when compared to the trees generated when the single general-purpose compressor was used. Additionally, when pairing the compression by column option and the *lzma* compressor, we were able to obtain better results than before, with every cluster apart from the ”pass” cluster being pure and a tree score of 0.99 as shown in Figure 4.4.

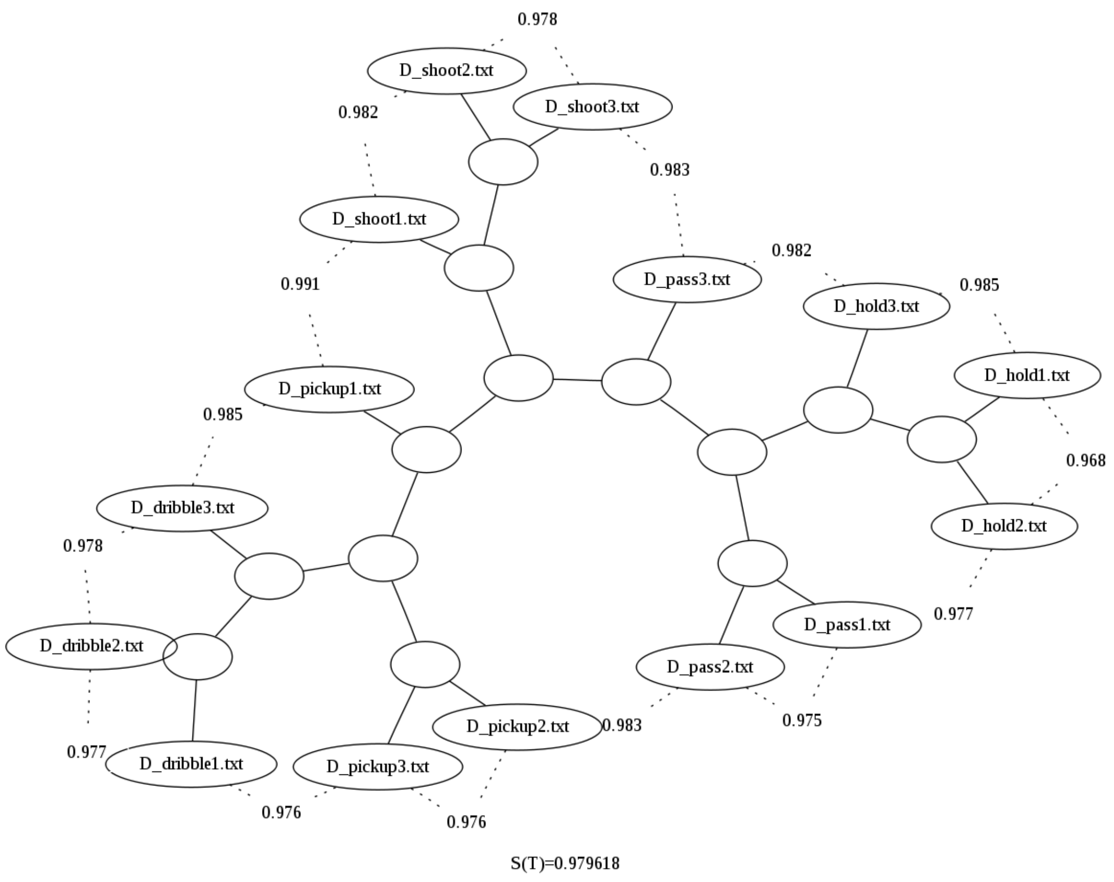


Figure 4.3: Clustering using *zlib* file sizes.

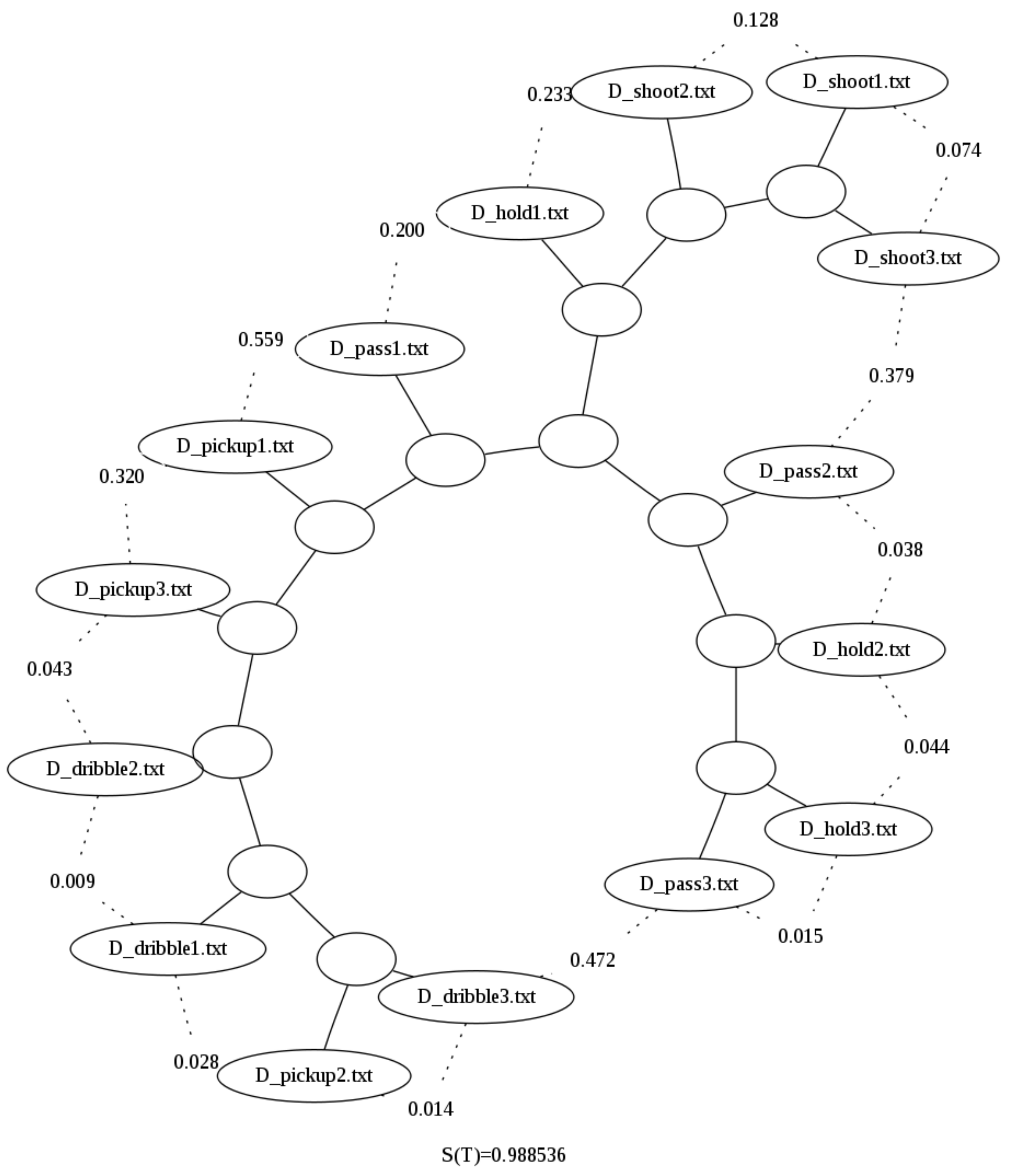


Figure 4.2: Binary-tree obtained from the raw file sizes.

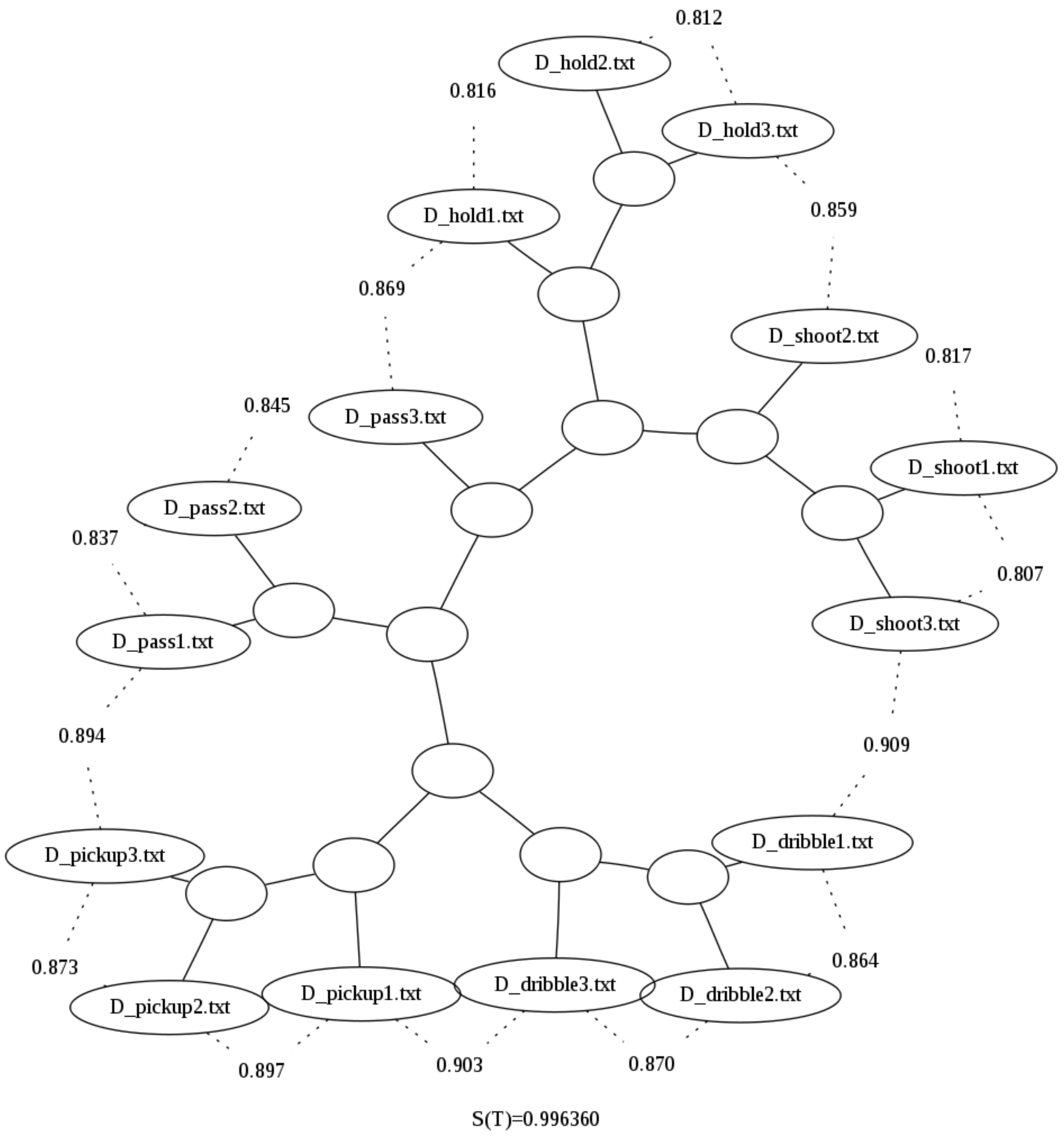


Figure 4.4: Clustering using *lzma* and compression by column.

Now, to answer the questions we started with, by applying the Complearn tool to multivariate/ time series data, and the results show to be consistent with what it is to be expected. Even so, these results can be improved if we use compression by column on our data. Also, after testing dedicated compressors such as the Delta and Gorilla compressors, we verified that they do not outperform the more generalist algorithms like *bzlib*, *zlib* and *lzma*.

Chapter 5

Data description and processing

5.1 Reuma.pt

This thesis is a result of the Project PREDICT (PTDC/CCI-CIF/29877/2017), funded by Fundo Europeu de Desenvolvimento Regional (FEDER), through Programa Operacional Regional LISBOA (LISBOA2020), and by national funds, through Fundação para a Ciência e Tecnologia (FCT). The Reuma.pt register has been active since June 2008. It was created by the Portuguese Society of Rheumatology, and it contains information received on a regular basis from rheumatic patients in Portugal who are performing biological therapy. In our study we will be focusing on SpA patients, but the Reuma.pt register includes patients with other conditions, such as rheumatoid arthritis, psoriatic arthritis, and juvenile idiopathic arthritis. Data from this repository includes identification data, demographic data, previous medical history, comorbidities, laboratory findings, past and current medications, adverse events, and disease activity, just to name a few.

5.2 Data Cleaning

We received our data On November 11, 2021. This data was provided in a 13 multi-sheet Excel file. Only patients with at least two appointments and known biological therapy are included in this database. The first two sheets contain 2293 rows, one per patient, and features related to patients' personal information, condition, and therapy. The next two sheets contain 32476 rows, one for each doctor's appointment, with data corresponding to the patient's joint health, disease indices as well as answers to questions used to assess general condition. The remaining nine sheets contain a variable number of rows and features per patient, with very noisy data, ambiguous features, and overall data we deemed not helpful for this work.

The Excel sheets pertaining to doctor's appointments and patient joints were chosen from among all the available data to be merged and used as the data for this work.

Each row in these sheets contains a patient id and an appointment sequence number,

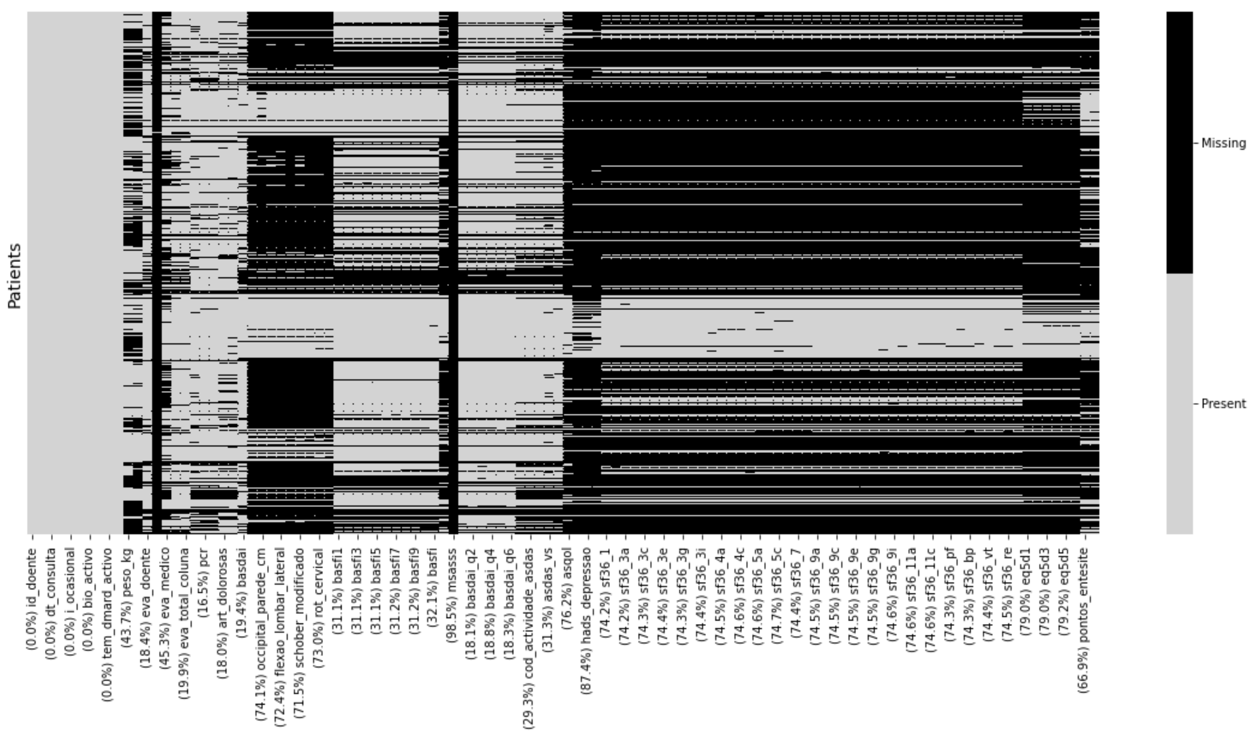


Figure 5.1: Data snippet of missing value volumes per column and patient

allowing a series of appointments to be set up for each patient. The number of appointments per patient is not fixed, and so we may have a different number of doctor's appointments for each patient. We produced two additional columns with the sum of all painful and swollen joints after merging these two datasets based on the aforementioned id and sequence number. Following a deeper examination, these were converted into binary columns due to the large disparity in the number of patients with no painful/swollen joints and those with one or more. The resulting dataset had 32476 rows and 112 columns.

We then focused on dealing with all the missing values in our dataset. A considerable portion of the columns in our dataset had massive percentages of missing values - see Figure 5.1.

The imputation of these values would not be a recommended approach since we certainly would be introducing a high bias in our data given the volume of value imputations that had to be done. For this reason, we chose to do a hybrid approach between dropping columns with a high volume of missing values followed by the elimination of rows with a high volume of missing values. For this approach, we started by dropping all columns with more than 33% of its values missing. We chose this threshold mainly because we get to keep all the disease indices (ASDAS, BASFI, and BASDAI). If this threshold were any lower, we would not be able to keep the BASFI index in the dataset. After dropping all the columns and rows, we ended up with a dataset comprised of 64 features and, 15804 rows. Patients in this dataframe are represented more than once, meaning every row in this dataset represents a doctor's appointment, but not all appointments are of a different patient.

Next, we created new columns we thought could better represent the information inside the already existing features. We created a chronological number for each of the patient's biological treatment, meaning every time the patient switched or stopped his treatment, this chronological number would restart. We added a column with the number of treatment switches at the time of each appointment, with the number increasing every time the patient switched treatment, but not when he stopped and restarted the same treatment. We also added some static features that were present in the patient's Excel sheet, that had relevant information about the patients' treatment journey, with these being the date of the first appointment, date of the last appointment and starting date of the first biological treatment. Finally, we dropped some of the columns that contained irrelevant information, these were *(*dt_consulta) for being a duplicate of other columns in the data, and *i_ocasional*, *i_ult_consulta* and *unidades_pcr* for being columns with a single value (and therefore contributing in nothing for the differentiation of the data).

5.3 Data Restructure

After dealing with all the missing variables and creating the new features, the next step was to structure our data such that it accurately represented the appointments of a single treatment as well as its success/failure outcome. When thinking of the best way to do this, while analyzing the appointments for each treatment in our data, the following pattern was found:

- The patient starts a biological treatment.
- When the patient has a high ASDAS, the next appointment shows great decreases in disease activity.
- After some time, the patient's activity increases slightly and stabilizes.

We believe this phenomenon occurs because, when patients have high disease activity, any treatment that has a minor impact on decreasing disease activity reflects a significant decrease in assessed ASDAS, most likely as a result of the patient finally experiencing a working treatment for his condition. As the treatment is continued, the patient becomes used to his new disease activity baseline, causing the assessed ASDAS levels to climb slightly.

Based on this hypothesis and the ASAS-EULAR recommendations, we considered representing a treatment by using the values from the first appointment at which the treatment is administered to the patient (treatment start) and computing the difference between the patient's ASDAS in the appointment and the one obtained in the last registered appointment (treatment end) for the same treatment. The resulting row contains all the characteristics that the patient had at the initial session, with the target variable being the ASDAS amplitude that results from this difference. After applying this transformation to our data, we end with a dataset consisting of 936 treatments instances (rows) and 64 features.

ASDAS amplitude between the first and last appointment can only be as big as the starting ASDAS. An improvement is considered clinically important when this amplitude is greater or equal to 1.1. These improvements are usually only obtained when the patient has high disease activity, corresponding to ASDAS greater than 2.1. For these reasons, it only makes sense to analyze the treatments with a starting ASDAS greater than 2.1 as recommended in by the ASAS-EULAR.

When we filter the dataset only to contain rows with these characteristics, we end with a dataset with 543 treatment instances (rows) and 64 features.

Finally, before separating our dataset into its different treatments, since the method we will be applying to our data is sensible to outliers, we remove them by taking the absolute Z-score of each value in the column relative to the column mean and standard deviation. We then consider only the rows where all column values have an absolute Z-score that is less than 3, eliminating any instances with values 3 standard deviations above the mean. The resulting dataframe without outliers is made up of 391 rows and 64 features. The final features are the following:

Patient id, appointment id, number of days being followed by a clinician, active biological treatment, sequential number of current biological treatment, concomitant disease-modifying antirheumatic drug (DMARD) at baseline, active corticoid treatment, patient evaluation during appointment, nocturnal patients evaluation, spine evaluation, C-reactive protein, vs, number of painful joints, number of swollen joints, answers to the BASFI questionnaire as well as the final BASFI index (11 features)), answers to the BASDAI questionnaire as well as the final BASDAI index (7 features), ASDAS index, ASDAS activity, ASDAS computed using vs, ASDAS computed using C reactive protein values, appointment count, chronological index of current biological treatment, number of switches up to the current appointment, age of disease diagnosis, estimated age of disease onset, birthdate, sex, nationality, first biological treatment, Anti-TNF index, baseline index, binary baseline indicator, Age at the beginning of first biological treatment, age ate the last appointment, disease years up to the last appointment, binary HLAB27 indicator, date of first appointment, date of last appointment, binary mortality indicator, patients alcoholism level, patients smoking level, time in years up to disease diagnosis, first failing treatment, years of disease up to first biological treatment and finally the target variable being patients clinical improvement.

Treatment	Row count
ADALIMUMAB	124
ETANERCEPT	120
GOLIMUMAB	67
INFLIXIMAB	64
SECUCINUMAB	11
CERTOLIZUMAB	4
USTECINUMAB	1

Table 5.1: Treatment row counts

Treatment	Treatment Success	Treatment Failure
ADALIMUMAB	62	62
ETANERCEPT	46	74
GOLIMUMAB	25	42
INFLIXIMAB	17	47

Table 5.2: Success and failure count per treatment

5.4 Final Datasets

Our final dataset consists of several patient biological therapies approaches. The final step in processing our data is to break it into its many biological therapies, so we can later apply the different clustering techniques discussed to the various treatments and determine which patient profiles best describe each of them. Each of the treatments has the following row count: The row count for each treatment can be seen in table 5.1.

Based on the row counts for each treatment, we decided to omit SECUCINUMAB, CERTOLIZUMAB, and USTECINUMAB from the dataset. Their sample size was so small that we believe an analysis of those treatments would be extremely skewed and not broad enough to accurately reflect population patterns.

With these treatment approaches excluded, we divided our dataset into its remaining treatments, with the final distribution of our target variable for each of these show in table 5.2.

Chapter 6

Clustering by compression results on Reuma.pt

In this chapter, we will look at the pipeline that was built and then applied to all the treatment datasets. We then examine each dataset separately, visualizing the impact of increasing the number of features on model scores, which compression method produced better results, and how clustering by compression models performed when compared to standard clustering models. The patterns discovered after examining the cluster feature distribution of the best 5 models for clustering by compression and the best 5 models for standard compression are then discussed. Finally, we compare the results obtained in each dataset to determine which trends were shared by all datasets and which were unique to each.

6.1 Methods

With all the data cleaned and separated into all the different treatments, we built a pipeline for the analysis of each dataset - see Figure 6.1. This pipeline has the ADALIMUMAB, ETANERCEPT, GOLIMUMAB and INFLIXIMAB as a starting point. We then perform feature selection for each dataset using the Maximum Relevance Minimum Redundancy (MRMR) algorithm, as explained in Section 3.6. After getting the best 10 features for each dataset we generate models starting by using the first two features with the best MRMR score, and appending the next best feature to the set until all the features were used to generate models. For each subset of features (2 best features, 3 best features, ..., 10 features) we also went over different numbers of clusters, from 2 to 10. Furthermore, for each pair of features and number of clusters we used Hierarchical Clustering (with complete, average and single distance) and K-medoids to generate 4 different models. Finally, when doing clustering by compression we used the 2 fastest compression algorithms, *zlib* and *bzlib* with both standard compression and compression by column.

This whole processes was done three times, once for clustering by compression with standard compression and other for normal clustering. This means that for each dataset we generate 1620 different models:

$$9(feature\ sets) \times 9(clusters) \times 4(models) \times 5(types\ of\ clustering) = 1620$$

with feature sets being 9 because we went from feature set of size 2 up to feature set of size 10, clusters being 9 because we went from 2 clusters up to 10 clusters, models being 4 because we used hierarchical clustering with 3 different distances and K-medoids, and types of clustering being 5 because we used *bzlib* and *zlib* with standard compression and compression by column, and normal clustering.

Each of these models was then evaluated using the silhouette score, v-measure and Adjusted Random Score, previously mentioned and explained in Section 3.7. After evaluating every model, we selected the 10 best models for each score, these being the top five models of clustering by compression and the top five models of standard clustering. These 10 models were then closely analyzed using their contingency matrices as to better understand and visualize their cluster distributions. The most pure and repeating clusters throughout all models were identified and had their features analyzed and interpreted so that common patient profiles for treatment failure and success could be extracted.

This pipeline was repeated an extra number of times for some special cases and tests. Firstly, the pipeline was run using the SMOTE [9] oversampling technique, which involves randomly duplicating examples from the minority class and adding them to the training dataset. We verified that a big portion of the patterns uncovered using this approach was highly biased, since the oversampling of the treatments implied that almost a quarter of the rows would be artificial when looking at the ETANERCEPT and GOLIMUMAB treatments and approximately 3 quarters of the rows would be artificial when looking at the INFLIXIMAB treatment. Undersampling was not an option either, since we already had a few rows to begin with, and we could not spare losing any more rows in order to balance the classes of treatment success and treatment failure. The other time we repeated the pipeline was to manually introduce features into the feature set of some treatments, in order to, in addition to performing the individual analysis of each treatment, be able to compare all treatments using the same set of features.

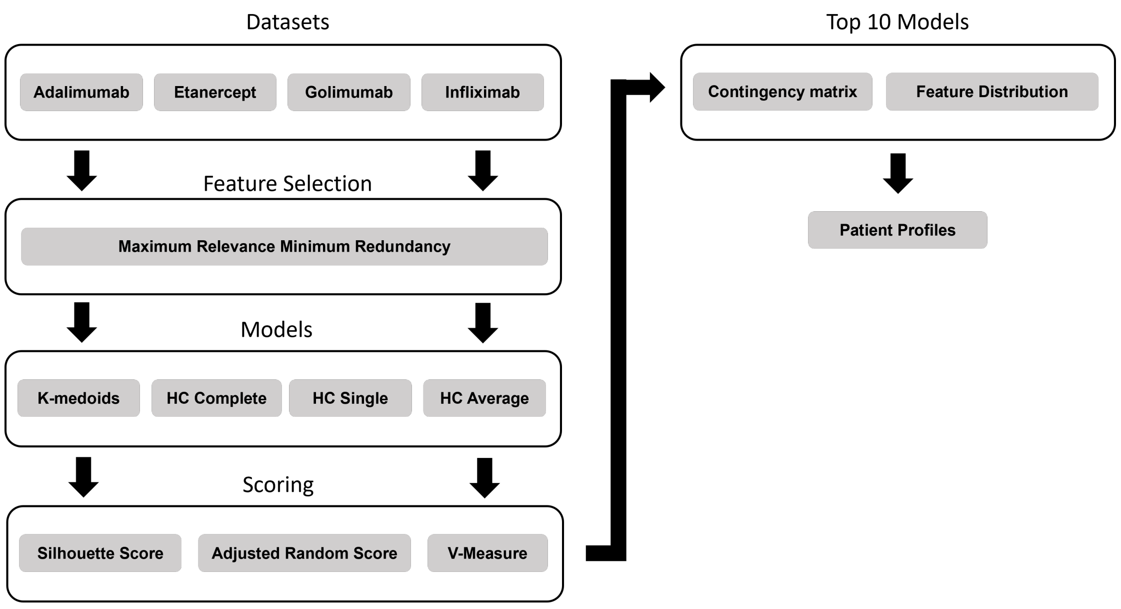


Figure 6.1: Flowchart representing the overall approach to the model building and data analysis done.

6.2 ADALIMUMAB

In this section, we will analyse the results obtained for the ADALIMUMAB dataset, which consists of a total of 124 treatments, 62 of them being success cases and the other 62 being failure cases.

The best 10 features selected by the MRMR method for this dataset were ASDAS, time until diagnostic, ASDAS calculated using pcr, number of days until first doctor's appointment, ASDAS calculated using vs, question six of the BASDAI questionnaire, vs, question four of the BASDAI questionnaire, BASDAI and pcr.

Following the feature selection, the models were generated. The best silhouette scores, v-measures and adjusted random scores for each number of features can be seen in Figure 6.2. Also, the best 5 clustering by compression models and the best 5 standard compression models for each of the scores can be seen in Tables 6.1, 6.2, 6.3.

When we used clustering by compression on the ADALIMUMAB dataset, we found that the compression by column models outperformed standard compression models for every score. Also, when comparing clustering by compression models and normal clustering models we see that the best models were created using clustering by compression were unable to beat the best models created using standard compression, but only when the models use a set of 2 features. For any other set of features greater than 2, clustering by compression was able to obtain competing scores and usually using a smaller number of clusters, meaning it was able to separate the data just as good as the standard clustering methods with fewer groups.

Score	Clusters	Features	Compressor	Model
0.873814	8	2	N/A	HC complete
0.671859	2	2	bzlib by column	HC average
0.671859	2	2	bzlib by column	HC complete
0.671859	2	2	bzlib by column	HC single
0.620176	8	2	N/A	HC average
0.579773	2	3	bzlib by column	HC single
0.579773	2	3	bzlib by column	HC average
0.537397	8	4	N/A	HC complete
0.513231	6	4	N/A	HC average
0.513231	6	4	N/A	HC single

Table 6.1: Silhouette Scores of the best 5 clustering by compression and the best 5 normal compression models for the ADALIMUMAB dataset.

Score	Clusters	Features	Compressor	Model
0.873814	8	2	N/A	HC complete
0.620176	8	2	N/A	HC average
0.513231	6	4	N/A	HC average
0.496547	5	4	N/A	HC complete
0.45292	7	4	bzlib by column	HC single
0.448966	5	4	bzlib by column	HC complete
0.435702	6	4	bzlib by column	HC average
0.422575	7	3	N/A	HC complete
0.373125	5	5	bzlib by column	HC complete
0.352834	8	5	bzlib by column	HC single

Table 6.2: V-measure of the best 5 clustering by compression and the best 5 normal compression models for the ADALIMUMAB dataset.

When looking at the feature distribution for the best models of each score and analyzing the feature distribution of their clusters, we discovered that the feature that best described treatment success or failure was the initial ASDAS, with patients having ASDAS greater than or equal to 3.2 having a higher rate of treatment success and patients having ASDAS less than 3.2 having a higher rate of treatment failure, with the patient gender playing no part in this distinction. We also verified that patients with ASDAS less than 2.7 have a higher treatment failure rate.

6.3 ETANERCEPT

In this section, we will analyze the results obtained for the ETANERCEPT dataset, which consists of a total of 120 treatments, 46 of them being success cases and the other 74 being failure cases.

Score	Clusters	Features	Compressor	Model
0.873814	8	2	N/A	HC complete
0.591835	7	2	N/A	HC average
0.539348	5	2	bzlib by column	HC average
0.535332	4	2	bzlib by column	HC complete
0.496547	5	4	N/A	HC complete
0.466472	5	4	N/A	HC average
0.444549	4	4	bzlib by column	HC complete
0.435702	6	4	bzlib by column	HC average
0.432731	6	4	bzlib by column	HC single
0.422575	7	3	N/A	HC complete

Table 6.3: Adjusted Random Scores of the best 5 clustering by compression and the best 5 normal compression models for the ADALIMUMAB dataset.

The best 10 features selected by the MRMR method for this dataset were ASDAS, time of disease until first biological treatment, ASDAS calculated using CRP (C-Reactive protein mg/l), number of days until first doctor's appointment, age at start of first biological treatment, question five of the BASDAI questionnaire, level of alcohol consumption, ASDAS calculated using ESR (erythrocyte sedimentation rate mm/hr), question three of the BASDAI questionnaire and CRP.

Following the feature selection, the same pipeline was applied, with the best silhouette scores, v-measures and adjusted random scores for each number of features being displayed in Figure 6.3. Also, the best 5 clustering by compression models and the best 5 standard compression models for each of the scores can be seen in Tables 6.4 6.5 6.6.

After applying clustering by compression to the ETANERCEPT dataset, we verified again that the compression by column outperformed the standard compression method for all scores. When comparing clustering by compression and standard compression, we observed again that the normal clustering had better maximum scores for small feature sets, of size two and three, and had similar scores to clustering by compression for feature sets greater than 3, with clustering by compression showing again a smaller number of clusters when compared to normal clustering.

Finally, when looking for patterns in the clusters of the best models, we verified that the features better describing patient profiles were ASDAS at the start of the treatment, time until first biological treatment and the patient sex. Patient with ASDAS less than 3.2 and time until first biological treatment less than 13 years, showed a high rate of treatment failure. For male patients with starting ASDAS greater than 3.2 the treatment showed a higher rate of success. We also verified again that patients with ASDAS less than 2.7 have a higher treatment failure rate.

Score	Clusters	Features	Compressor	Model
0.513297	7	3	N/A	HC single
0.513297	7	3	N/A	HC average
0.513297	7	3	N/A	HC complete
0.511218	7	3	bzlib by column	HC single
0.510276	6	3	bzlib by column	HC average
0.510276	6	3	bzlib by column	HC complete
0.430986	2	9	bzlib normal	HC single
0.415792	2	7	bzlib normal	HC single
0.393228	7	4	N/A	HC complete
0.393228	7	4	N/A	HC average

Table 6.4: Silhouette Scores of the best 5 clustering by compression and the best 5 standard compression models for the ETANERCEPT dataset.

Score	Clusters	Features	Compressor	Model
0.507932	8	3	N/A	HC average
0.499001	8	3	N/A	HC complete
0.490582	8	3	N/A	HC single
0.3945	8	2	bzlib by column	HC average
0.393228	7	4	N/A	HC complete
0.382533	2	3	bzlib by column	HC single
0.382533	2	3	bzlib by column	HC complete
0.378066	8	2	N/A	HC average
0.318598	2	4	bzlib by column	HC complete
0.290999	4	2	bzlib by column	HC complete

Table 6.5: V-measures of the best 5 clustering by compression and the best 5 standard compression models for the ETANERCEPT dataset.

Score	Clusters	Features	Compressor	Model
0.507932	8	3	N/A	HC average
0.499001	8	3	N/A	HC complete
0.404663	6	3	N/A	HC single
0.378066	8	2	N/A	HC average
0.37364	8	2	N/A	HC complete
0.353129	6	2	bzlib by column	HC average
0.290999	4	2	bzlib by column	HC complete
0.256796	2	6	bzlib by column	HC average
0.256255	5	3	zlib by column	HC average
0.256255	5	3	zlib by column	HC single

Table 6.6: Adjusted Random Scores of the best 5 clustering by compression and the best 5 standard compression models for the ETANERCEPT dataset.

Score	Clusters	Features	Compressor	Model
0.588178	8	2	N/A	HC complete
0.583808	8	2	N/A	HC average
0.56795	2	3	bzlib by column	HC average
0.56795	2	3	bzlib by column	K-medoids
0.56795	2	3	bzlib by column	HC complete
0.56795	2	3	bzlib by column	HC single
0.505713	8	4	bzlib by column	HC complete
0.475152	8	4	N/A	HC complete
0.446612	7	4	N/A	HC average
0.441733	6	4	N/A	HC single

Table 6.7: Silhouette scores of the best 5 clustering by compression and the best 5 standard compression models for the GOLIMUMAB dataset.

6.4 GOLIMUMAB

In this section, we will analyze the results obtained for the GOLIMUMAB dataset, which consists of a total of 67 treatments, 25 of them being success cases and the other 42 being failure cases.

The best 10 features selected by the MRMR method for this dataset were ASDAS, time until diagnosis, sex, ASDAS calculated using pcr, Date of the last doctor's appointment, patient's evaluation, number of treatment switches, age at condition diagnosis, pcr, baseline index.

After the feature selection process, we applied the pipeline, with the resulting best silhouette scores, v-measures and adjusted random scores for each number of features being displayed in Figure 6.4. Also, the best 5 clustering by compression models and the best 5 standard compression models for each of the scores can be seen in Tables 6.7 6.8 6.9.

We then applied the pipeline to the GOLIMUMAB dataset and verified very similar score results between clustering by compression models and normal clustering models. The trend of clustering by compression, needing fewer clusters to separate the data with similar scores when compared with the normal clustering models, continues for this dataset.

As for the patterns identified in the clusters, we verified that male patients with ASDAS greater than 2.7 have a higher treatment success rate. In contrast, most of the female patients (26 out of 34 female patients) had treatment failure. Also, like what we verified before, patients with ASDAS less than 2.7 have a higher rate of treatment failure.

Score	Clusters	Features	Compressor	Model
0.588178	8	2	N/A	HC complete
0.505713	8	4	bzlib by column	HC complete
0.502198	8	4	bzlib by column	HC single
0.475152	8	4	N/A	HC complete
0.450261	8	5	bzlib by column	HC complete
0.446612	7	4	N/A	HC average
0.425628	8	5	N/A	HC complete
0.390852	3	3	bzlib by column	K-medoids
0.390682	8	5	bzlib by column	HC single
0.386077	8	5	N/A	HC average

Table 6.8: V-measures of the best 5 clustering by compression and the best 5 standard compression models for the GOLIMUMAB dataset.

Score	Clusters	Features	Compressor	Model
0.446612	7	4	N/A	HC complete
0.441733	6	4	N/A	HC single
0.432458	8	4	N/A	HC average
0.412846	3	3	bzlib by column	HC complete
0.407364	7	5	N/A	HC complete
0.390852	3	3	bzlib by column	K-medoids
0.390682	8	5	bzlib by column	HC single
0.386953	5	5	N/A	HC average
0.374891	3	5	bzlib by column	HC complete
0.308163	3	3	zlib by column	HC average

Table 6.9: Adjusted Random Scores of the best 5 clustering by compression and the best 5 standard compression models for the GOLIMUMAB dataset.

Score	Clusters	Features	Compressor	Model
0.8125	8	2	N/A	HC average
0.803819	8	2	N/A	HC complete
0.684863	8	2	bzlib by column	HC single
0.679641	2	2	bzlib by column	HC complete
0.679641	2	2	bzlib by column	HC average
0.60313	8	2	N/A	HC single
0.495979	7	2	bzlib by column	K-medoids
0.451636	2	3	bzlib by column	HC single
0.278344	5	5	N/A	HC average
0.259509	5	5	N/A	HC complete

Table 6.10: Silhouette Scores of the best 5 clustering by compression and the best 5 standard compression models for the INFLIXIMAB dataset.

6.5 INFLIXIMAB

In this section, we will analyze the results obtained for the INFLIXIMAB dataset, which consists of a total of 64 treatments, 17 of them being success cases and the other 47 being failure cases.

After applying the MRMR feature selection algorithm the best 10 features extracted were ASDAS, Sex, question four of the BASDAI questionnaire, question two of the BASDAI questionnaire, ASDAS calculated using pcr, age at start of first biologic treatment, question three of the BASFI questionnaire, vs, BASDAI and number of treatment switches.

We then generated models with all the features selected, with the best scores being shown in Figure 6.5. The best 5 clustering by compression models and the best 5 standard compression models that we used for the treatment analysis have their scores shown in Tables 6.10 6.11 6.12.

Following the model generation and selection, we verified similar results to the one we encountered in the previous datasets, where the normal clustering outperforms clustering by compression for feature sets of size two, but has very similar results for any other feature set sizes. Clustering by compression shows smaller number of clusters when compared to normal clustering.

Looking at the patient profiles, we observed that male patients with initial ASDAS greater than 3.2 had a higher treatment success rate for this biological treatment. Female patients failed at a greater incidence for any level of initial ASDAS. Finally, patients with an initial ASDAS of less than 2.7 had a higher failure rate, making it consistent with all the previous treatments.

Score	Clusters	Features	Compressor	Model
0.8125	8	2	N/A	HC average
0.803819	8	2	N/A	HC complete
0.684863	8	2	bzlib by column	HC single
0.669274	8	2	bzlib by column	HC complete
0.669274	8	2	bzlib by column	HC average
0.60313	8	2	N/A	HC single
0.495979	8	2	bzlib by column	K-medoids
0.292292	3	4	bzlib normal	HC average
0.266225	6	5	N/A	HC average
0.244656	8	3	N/A	HC average

Table 6.11: V-measures of the best 5 clustering by compression and the best 5 standard compression models for the INFLIXIMAB dataset.

Score	Clusters	Features	Compressor	Model
0.8125	8	2	N/A	HC average
0.803819	8	2	N/A	HC complete
0.513128	4	2	bzlib by column	HC average
0.513128	4	2	bzlib by column	HC complete
0.303977	2	7	bzlib normal	HC single
0.300028	5	3	bzlib by column	HC complete
0.296687	8	2	zlib by column	HC single
0.284965	5	2	N/A	HC single
0.217403	3	3	N/A	HC average
0.175645	6	3	N/A	HC complete

Table 6.12: Adjusted Random Scores of the best 5 clustering by compression and the best 5 standard compression models for the INFLIXIMAB dataset.

6.6 Comparison of results

In this section, we will focus on the comparison of the results obtained for all the different treatments previously analyzed, and extract overall conclusions from these results.

When looking at the results from all the datasets, we could observe that the majority of the patterns could be seen in all the datasets. Starting by the things all treatments had in common:

- Clustering by compression using compression by column showed better results for all the biological treatments.
- Clustering by compression demonstrated the ability to achieve similar score results to standard clustering for feature sets greater than 2, while using a smaller number of clusters, implying that it was able to separate the data just as well as normal clustering models without dividing it into as many clusters.
- The results of the INFLIXIMAB, GOLIMUMAB and ETANERCEPT treatments, showed that for this dataset, the treatments had a higher success rate for male patients whereas the improvements in female patients wasn't as noticeable.
- The higher the initial ASDAS the higher the success rate for the treatment applied.
- When the initial ASDAS was lower than 2.7 (and therefore less room for an improvement), treatments are more likely to be classified as ”failures”, since ASDAS improvements, although sometimes present, didn't exceed the 1.1 ASDAS reduction threshold defined by ASAS-EULAR recommendations.

For the results not found in all datasets, we found one pattern specific to the GOLIMUMAB dataset:

- In contrast to the INFLIXIMAB, GOLIMUMAB and ETANERCEPT treatments, the ADALIMUMAB did not make a distinction between genders, having similar success rates for both.

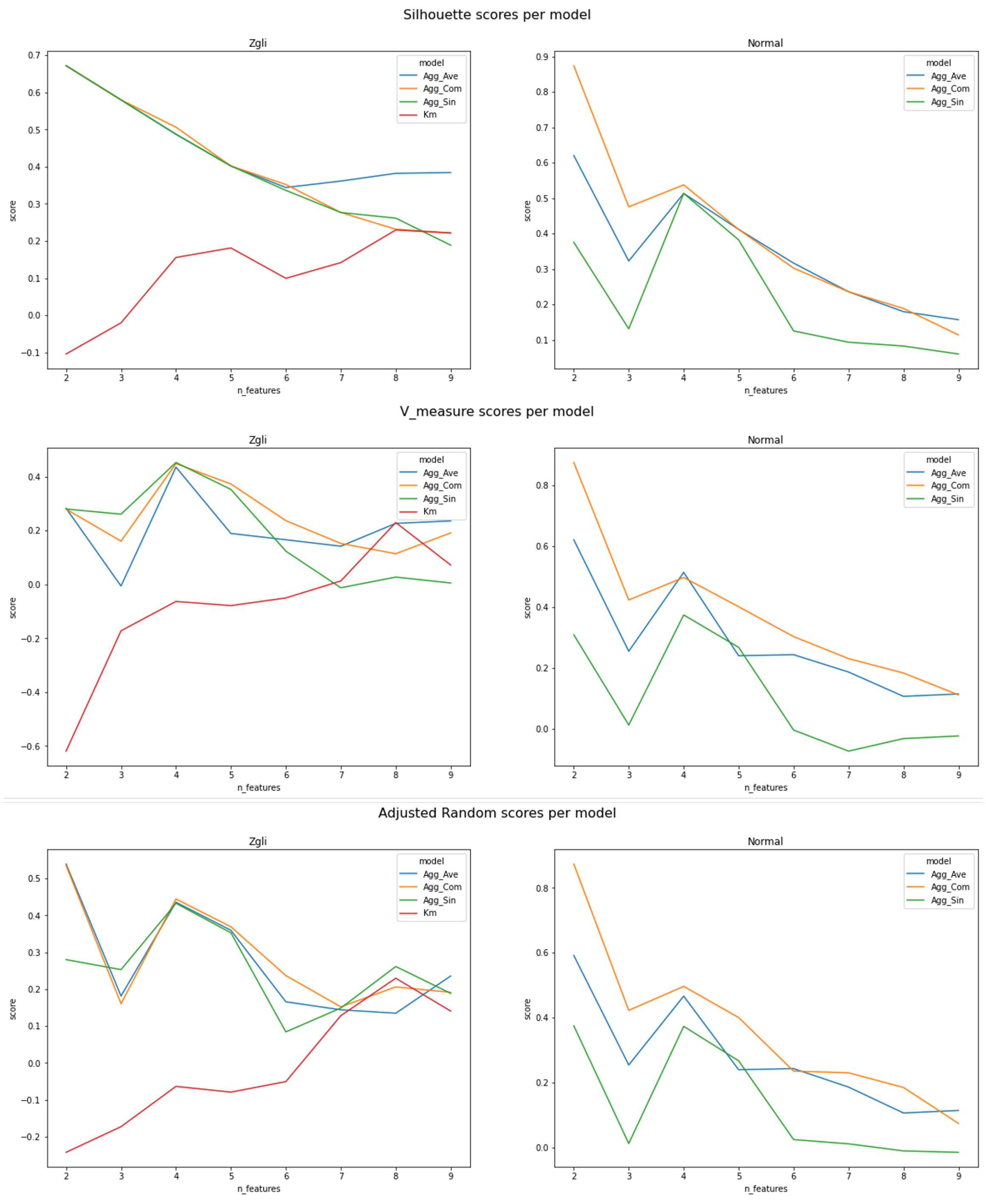


Figure 6.2: Best scores per number of features for the ADALIMUMAB dataset.

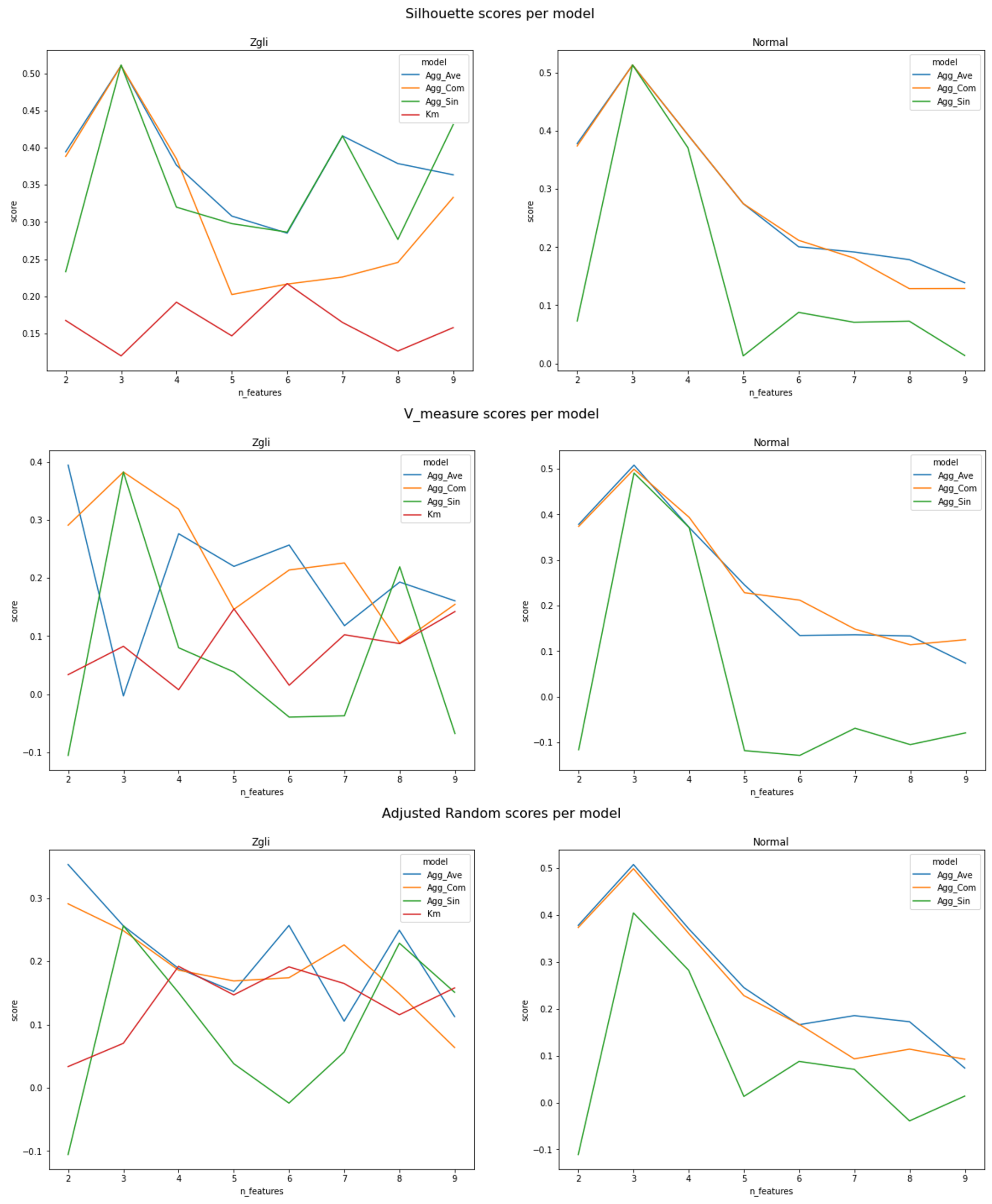


Figure 6.3: Best scores per number of features for the ETANERCEPT dataset

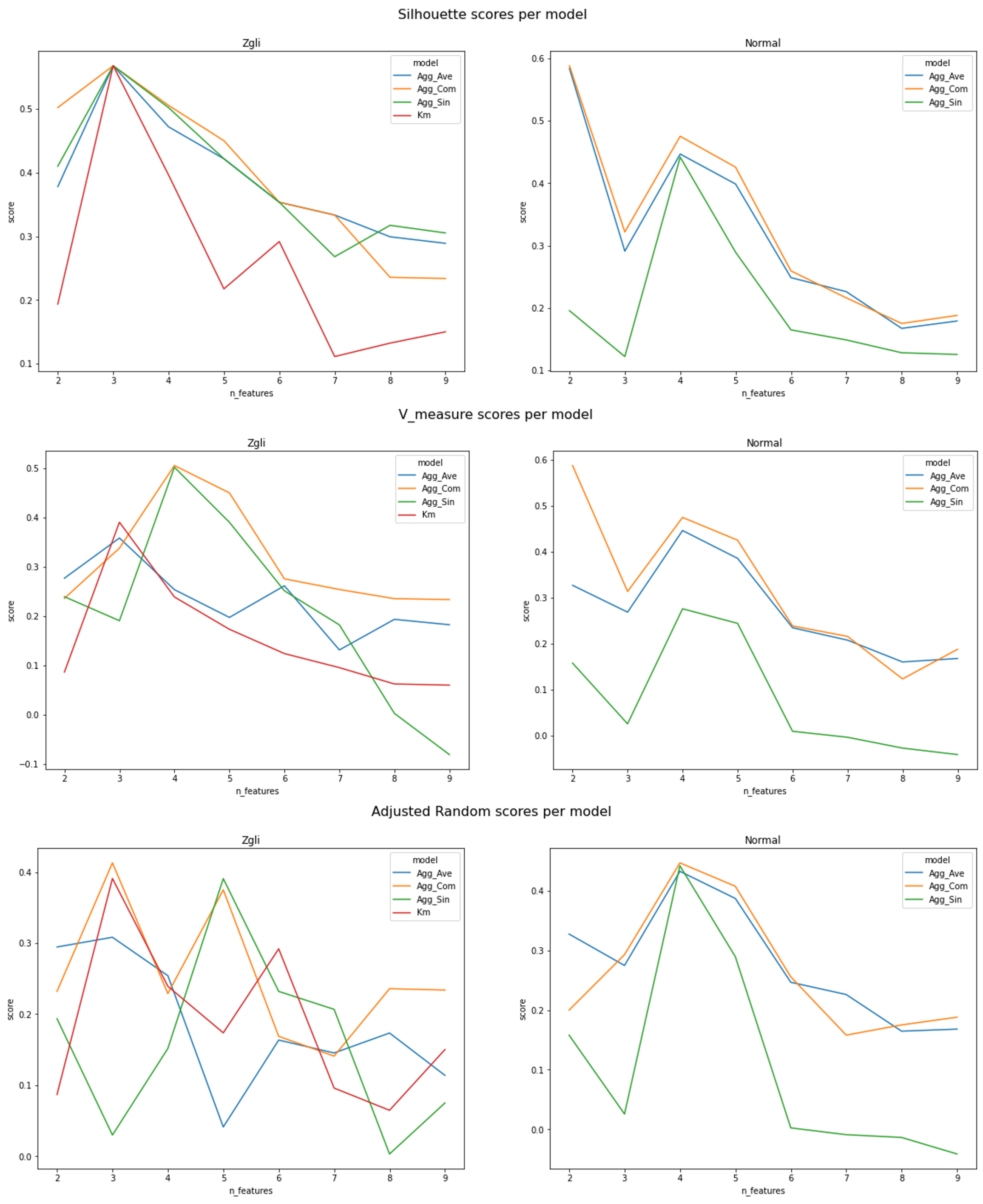


Figure 6.4: Best scores per number of features for the GOLIMUMAB dataset.

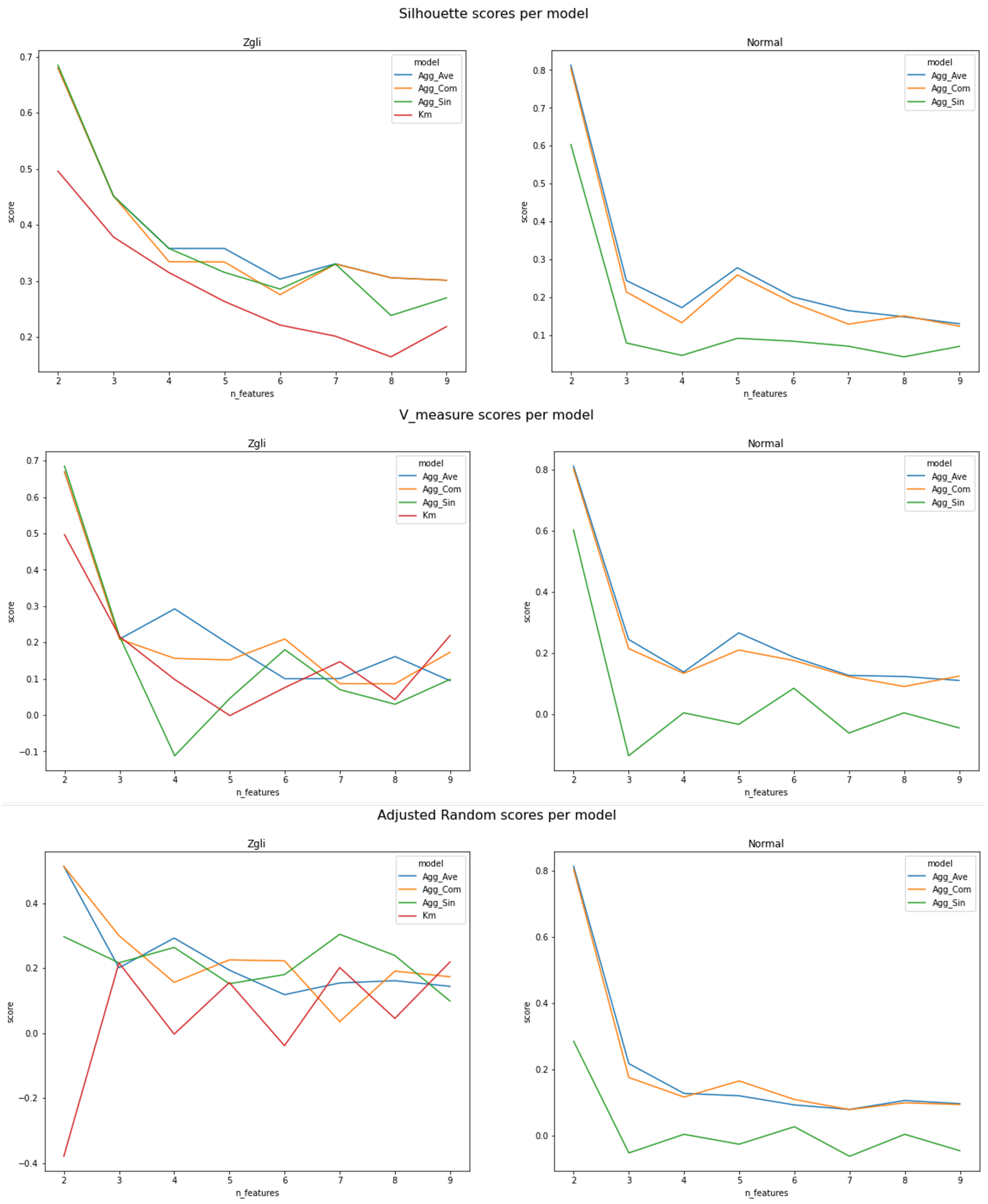


Figure 6.5: Best scores per number of features for the INFLIXIMAB dataset.

Chapter 7

Conclusions

Clustering by compression is an unconventional method of grouping and mining data for patterns. This method was chosen for mining the Reuma.pt dataset for patterns that could link patient profiles to different biological treatments in this study and find which treatment could better suit each patient.

Conclusions can be drawn at two levels based on the work done and the results achieved. First, we examined clustering by compression and the various conclusions found in this work. When developing and testing the Zgli python module, we verified that text compressors such as *bzlib*, *zlib* and *lzma* generated better models than specific compression methods such as delta for integer compression and gorilla for float compression.

Models constructed on a distance matrix computed using the compression by column method consistently outperformed models built using the standard compression method for both the basket and Reuma.pt datasets. This leads to the conclusion that when working with tabular data, the compression per column method can better preserve the structure of the data, resulting in a less noisy and more representative compressed data sizes.

Second, when we examined the patterns between SpA patients and the various biologic treatments, we could clearly see that, for the dataset obtained at the end of data pre-processing, the INFLIXIMAB, GOLIMUMAB, and ETANERCEPT treatments had a higher success rate for male patients and a higher failure rate for female patients, with the ADALIMUMAB treatment being the only exception, with similar success rates for both male and female patients. We could also see that the greater the initial ASDAS for all treatments, the higher the rate of treatment success. The contrary was also seen, with the higher the rate of treatment failure being reported when the baseline ASDAS was closer to the 2.1 to 2.7 range. Because the concept of treatment success and failure is based on ASDAS variance at 12 weeks, we believe that when we look at patients with high initial ASDAS, and hence high disease activity, patients are particularly sensitive to any decreases in disease activity, that could lead to an improvement in their quality of life. The same could be true for patients with initial ASDAS values ranging from 2.1 to 2.7. While these individuals have disease activity that is classified as high, their initial ASDAS is near to the 2.1 threshold, with patients with ASDAS less than 2.1 being classified as having low disease activity. The lower the disease activity, the more difficult it is to find treatments that have a positive effect on the reduction of disease activity.

From the results observed, for male patients with high initial ASDAS, any of the four biologic treatments analyzed (INFLIXIMAB, GOLIMUMAB, ETANERCEPT and ADALIMUMAB) showed to be good approach to obtain clinical important improvements on disease activity. For female patients with high initial ASDAS, ADALIMUMAB showed to be the better approach to obtain clinical important improvements. In the case of patients with initial ASDAS values ranging between 2.1 and 2.7, we were unable to identify any relevant patterns that linked patients to treatment success, since success instances were scarce. We believe that when looking at the treatments SECUCINUMAB, CERTOLIZUMAB and USTECINUMAB as well as patients between the ASDAS range of 2.1 and 2.7, more data is needed before any meaningful patient profiles representative of the larger population can be found, and respective conclusions can be taken.

7.1 Achievements

Throughout this work we developed and made available Zgli [7] a python module with different tools that allow its users to perform the computation of distance matrices comprised of the normalized compression distance between the different objects to be clustered. This tool expands the number of compression algorithms that can be used for compression, provides a compression by column option for when the user is dealing with tabular data, and also provides a data encoder for small data that would otherwise be incompatible with the clustering by compression methodology.

A dataset comprised patient treatment instances of 4 different biological treatments, INFLIXIMAB, GOLIMUMAB, ETANERCEPT and ADALIMUMAB was created and can be used for future studies on SpA patients from the Reuma.pt data repository.

Finally, different patient profiles were identified that linked patients with different successful and unsuccessful treatment approaches, that could bring valuable insights for aiding clinicians in selection of the right biologic treatment for each patient.

7.2 Future work

Considering the work developed during this thesis and some of the results obtained, we identified the following proposals for future work.

Firstly, we think that the phenomena of clustering by compression needing fewer clusters to obtain similar scores when compared to normal clustering models should be further explored by applying this same comparative approach to other datasets comprised of tabular data and see if the same result can be observed.

We also see that there is more potential when it comes to different data encoding and compression approaches, with the conversion of text data into different formats such as images and sound allowing us to explore several different compression approaches that could produce radically different results due to the significant difference in the representation of the data.

Finally, the quartet method could be added to the Zgli module to enable users to perform the same approaches discussed in the initial literature published in clustering by compression.

Bibliography

[1] 2.3. clustering.

[2] Calin A, Garrett S, Whitelock H, Kennedy L G, O'Hea J, Mallone P, and Jenkinson T. Bath ankylosing spondylitis functional index test, 2005.

[3] Deepak Agnihotri, Kesari Verma, and Priyanka Tripathi. Pattern and cluster mining on text data. In *2014 Fourth International Conference on Communication Systems and Network Technologies*, pages 428–432, 2014.

[4] Christopher P. Ames, Justin S. Smith, Ferran Pellisé, Michael Kelly, Ahmet Alanay, Emre Acaroğlu, Francisco Javier Pérez-Grueso, Frank Kleinstück, Ibrahim Obeid, Alba Vila-Casademunt, and et al. Artificial intelligence based hierarchical clustering of patient types and intervention categories in adult spinal deformity surgery. *Spine*, 44(13):915–926, 2019.

[5] Calin Andrei, Garrett S, Jenkinson T, Kennedy L G, Whitelock H, and Gaisford P. A new approach to defining functional ability in ankylosing spondylitis: The development of the bath ankylosing spondylitis functional index. *Classic Papers in Rheumatology*, 21:2281–5, 1994.

[6] Iben Axén, Lennart Bodin, Gunnar Bergström, Laszlo Halasz, Fredrik Lange, Peter W Lövgren, Annika Rosenbaum, Charlotte Leboeuf-Yde, and Irene Jensen. Clustering patients on the basis of their individual course of low back pain over a six month period. *BMC Musculoskeletal Disorders*, 12(1), 2011.

[7] Diogo Azevedo and André Souto. Import zgli a clustering technique., Feb 2022.

[8] Carolina Barata, Ana Maria Rodrigues, Helena Canhão, Susana Vinga, and Alexandra Carvalho. Predicting biologic therapy outcome of patients with spondyloarthritis: Joint models for longitudinal and survival analysis. *JMIR Med Inform*, 9(7):e26823, Jul 2021.

[9] Rok Blagus and Lara Lusa. Smote for high-dimensional class-imbalanced data - bmc bioinformatics, Mar 2013.

[10] Jaime Branco, Ana Rodrigues, Nélia Gouveia, Mónica Eusébio, Sofia Ramiro, Pedro Machado, Leonor Pereira da Costa, Ana Filipa Mourão, Inês Silva, Pedro Laires, Alexandre Sepriano, Filipe Araújo, Sónia Gonçalves, Pedro Coelho, Viviana Tavares, Jorge Cerol, Jorge Mendes, Loreto Carmona, Helena Canhão, and on behalf of the EpiReumaPt study group. Prevalence of rheumatic and musculoskeletal diseases and their impact on health-related quality of life, physical function and mental health in portugal: results from epireumapt– a national health survey. *RMD Open*, 2(1), 2016.

[11] A Calin, S Garrett, H Whitelock, L G Kennedy, J O'Hea, P Mallorie, and T Jenkinson. A new approach to defining functional ability in ankylosing spondylitis: the development of the bath ankylosing spondylitis functional index. *J. Rheumatol.*, 21(12):2281–2285, December 1994.

[12] Rudi Cilibrasi, Anna Cruz, and Steven Rooij. Complearn, Feb 2008.

[13] Rudi Cilibrasi and Paul Vitányi. Clustering by compression. *IEEE Transactions on Information theory*, 51(4):1523–1545, 2005.

[14] Rudi Cilibrasi and Paul Vitányi. Phylogeny of the covid-19 virus sars-cov-2 by compression, 2020.

[15] Rudi Cilibrasi, Paul Vitanyi, and Ronald Wolf. Algorithmic clustering of music. In *Proceedings of the Fourth International Conference on Web Delivering of Music, 2004. EDELMUSIC 2004.*, pages 110–117, Spain, 2004. IEEE.

[16] Chris Ding and Hanchuan Peng. Minimum redundancy feature selection from microarray gene expression data.

[17] John Ellson, Emden Gansner, Yifan Hu, Stephen North, Magnus Jacobsson, Matthew Fernandez, Mark Hansen, Vladimir Alexiev, Arif Bilgin, Don Caldwell, and et al., Aug 2021.

[18] Pedregosa Fabian, Varoquaux Gael, Gramfort Alexandre, Michel Vincent, Thirion Bertrand, Grisel Oliver, Blondel Mathieu, Prettenhofer Peter, Weiss Ron, Dubourg Vincent, and et al. 2.3. clustering, 2007.

[19] Ernst Feldtkeller, Ernst Lemmel, and Anthony Russell. Ankylosing spondylitis in the pharaohs of ancient egypt. *Rheumatology international*, 23:1–5, 02 2003.

[20] Ernst Feldtkeller, Ernst-Martin Lemmel, and Anthony Russell. Ankylosing spondylitis in the pharaohs of ancient egypt. *Rheumatology international*, 23(1):1–5, 2003.

[21] S Garrett, T Jenkinson, L G Kennedy, H Whitelock, P Gaisford, and A Calin. A new approach to defining disease status in ankylosing spondylitis: the bath ankylosing spondylitis disease activity index. *J. Rheumatol.*, 21(12):2286–2291, December 1994.

[22] David Guarin, Juan Gloria, and Leonel Naranjo. Basketball dataset data set, 2019.

[23] Marc C. Hochberg, Ellen M. Gravallese, Alan J. Silman, Josef S. Smolen, Michael E. Weinblatt, and Michael H. Weisman. *Rheumatology*. Elsevier, 7 edition, 2019.

[24] Barbara Kean, Cathy Garrard, Cathy Cassata, Joseph Bennington-Castro, Erica Patino, Nina Wasserman, Beth Levine, and Emilia Benton. What are rheumatic diseases? symptoms, causes, diagnosis, treatment, and prevention, Sep 2020.

[25] Ming Li, Jonathan Badger, Xin Chen, Sam Kwong, Paul Kearney, and Haoyong Zhang. An information-based sequence distance and its application to whole mitochondrial genome phylogeny . *Bioinformatics*, 17(2):149–154, 02 2001.

[26] Ming Li, Xin Chen, Xin Li, Bin Ma, and Paul Vitanyi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, 2004.

[27] Joshua Lockerman and Ajay Kulkarni. Time-series compression algorithms, explained, Mar 2021.

[28] Pedro M Machado, Robert Landewé, and Désirée van der Heijde. Ankylosing spondylitis disease activity score (asdas): 2018 update of the nomenclature for disease activity states. *Annals of the Rheumatic Diseases*, 77(10):1539–1540, 2018.

[29] Farrouq Mahmood and Philip Helliwell. Ankylosing spondylitis: A review. *EMJ rheumatology*, 2(4):134–139, Aug 2017.

[30] E Martín-Mola, J Sieper, M Leirisalo-Repo, B A C Dijkmans, B Vlahos, R Pedersen, A S Koenig, and B Freundlich. Sustained efficacy and safety, including patient-reported outcomes, with etanercept treatment over 5 years in patients with ankylosing spondylitis. *Clin. Exp. Rheumatol.*, 28(2):238–245, March 2010.

[31] Samuele Mazzanti. ”mrmr” explained exactly how you wished someone explained to you, Feb 2022.

[32] Wes McKinney. Pandas, Jan 2008.

[33] Ankur A Patel. *Hands-on unsupervised learning using Python: how to build applied machine learning solutions from unlabeled data*. O’Reilly Media, 2019.

[34] Tuomas Pelkonen, Scott Franklin, Paul Cavallaro Justin Teller, Qi Huang, Justin Meza, and Kaushik Veeraraghavan. Gorilla: A fast, scalable, in-memory time series database, Mar 2021.

[35] Kishan Rama, Helena Canhão, Alexandra Carvalho, and Susana Vinga. AliClu - temporal sequence alignment for clustering longitudinal clinical data. *BMC Medical Informatics and Decision Making*, 19(1):289, December 2019.

[36] Sofia Ramiro, Pedro Machado, Raquel Roque, Helena Santos, Joaquim Polido-Pereira, Daniela Peixoto, Catia Duarte, Fernando Pimentel-Santos, Candida Silva, João Fonseca, et al. Predictive factors of response at 12 weeks in patients with ankylosing spondylitis starting biological therapies—results from the portuguese register—reuma. pt. *Clin Exp Rheumatol*, 30:641–2, 2012.

[37] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 410–420, 2007.

[38] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[39] Costa Santos, Joao Bernardes, Paul Vitányi, and Luis Antunes. Clustering fetal heart rate tracings by compression. In *19th IEEE Symposium on Computer-Based Medical Systems (CBMS'06)*, pages 685–690, USA, 2006. IEEE.

[40] Editorial Team and Emily Downward. What are the symptoms of as?, Feb 2019.

[41] Carme Uribe, Barbara Segura, Hugo Cesar Baigo, Alexandra Abos, Anna Isabel Garcia-Diaz, Anna Campabadal, Maria Jose Marti, Francesc Valldeoriola, Yaroslau Compta, Eduard Tolosa, and et al., Feb 2018.

[42] D van der Heijde, E Lie, T K Kvien, J Sieper, F Van den Bosch, J Listing, J Braun, R Landewé, and Assessment of SpondyloArthritis international Society (ASAS). ASDAS, a highly discriminatory ASAS-endorsed disease activity score in patients with ankylosing spondylitis. *Ann. Rheum. Dis.*, 68(12):1811–1818, December 2009.

[43] Désirée van der Heijde, Sofia Ramiro, Robert Landewé, Xenofon Baraliakos, Filip Van den Bosch, Alexandre Sepriano, Andrea Regel, Adrian Ciurea, Hanne Dagfinrud, Maxime Dougados, Floris van Gaalen, Pál Géher, Irene van der Horst-Bruinsma, Robert D Inman, Merryn Jongkees, Uta Kiltz, Tore K Kvien, Pedro M Machado, Helena Marzo-Ortega, Anna Molto, Victoria Navarro-Compàn, Salih Ozgocmen, Fernando M Pimentel-Santos, John Reveille, Martin Rudwaleit, Jochen

Sieper, Percival Sampaio-Barros, Dieter Wiek, and Jürgen Braun. 2016 update of the ASAS-EULAR management recommendations for axial spondyloarthritis. *Ann. Rheum. Dis.*, 76(6):978–991, June 2017.